A Continuous Silent Speech Recognition System for AlterEgo, a Silent Speech Interface

by

Eric J. Wadkins

B.S., Massachusetts Institute of Technology (2018)

Submitted to the
Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2019

Author . . . .
Department of Electrical Engineering and Computer Science
May 24, 2019

Certified by . . . .
Pattie Maes
Professor, Media Arts and Sciences
Thesis Supervisor

Accepted by . . . .
Katrina LaCurts
Chair, Master of Engineering Thesis Committee

A Continuous Silent Speech Recognition System for AlterEgo, a Silent Speech Interface

by

Eric J. Wadkins

Submitted to the Department of Electrical Engineering and Computer Science
on May 24, 2019, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

In this thesis, I present my work on a continuous silent speech recognition system for AlterEgo, a silent speech interface. By transcribing residual neurological signals sent from the brain to speech articulators during internal articulation, the system allows one to communicate without the need to speak or perform any visible movements or gestures. It is capable of transcribing continuous silent speech at a rate of over 100 words per minute. The system therefore provides a natural alternative to normal speech at a rate not far below that of conversational speech. This alternative method of communication enables those who cannot speak, such as people with speech or neurological disorders, as well as those in environments not suited for normal speech, to communicate more easily and quickly. In the same capacity, it can serve as a discreet, digital interface that augments the user with information and services without the use of an external device.

I discuss herein the data processing and sequence prediction techniques used, describe the collected datasets, and evaluate various models for achieving such a continuous system, the most promising among them being a deep convolutional neural network (CNN) with connectionist temporal classification (CTC). I also share the results of various feature selection and visualization techniques, an experiment to quantify electrode contribution, and the use of a language model to boost transcription accuracy by leveraging the context provided by transcribing an entire sentence at once.

Thesis Supervisor: Pattie Maes
Title: Professor, Media Arts and Sciences

Acknowledgments

I am privileged to have had the opportunity to think, work, and learn surrounded by so many incredible people at MIT. They will continue to serve as an inspiration to me.

I would like to thank my advisor, Prof. Pattie Maes, and project lead, Arnav Kapur, as well as so many others in the Media Lab's Fluid Interfaces Group, for providing guidance and mentorship throughout the course of my Master's program.

Finally, I would like to thank my family and friends for their continued support this past year and throughout my entire time at MIT. It would not have been possible without them.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

As humans, we have an incredible ability to understand information presented to us. We are fortunate for this, as information plays a fundamental role in our lives. Our decision making process is driven by access to information. And communication, the means through which we share knowledge and form relationships with others, is, at its simplest, the transfer of information. For this reason, access to information (or lack thereof) is a limiting factor in what we can achieve, and can potentially interfere with our ability to problem solve and communicate. Internet-connected computing devices, on the other hand, have unmatched access to information, as well as the computing power to solve many problems humans do not have the capacity to solve. Still, machines cannot yet replace humans in performing many tasks. Therefore, a natural solution arises – the coupling of human and machine, using the strengths of both to more effectively achieve the user's goals.

AlterEgo is a silent speech interface that aims to augment a human user with the information and computing power available to a computing device, while also providing an alternative means of communication. It is a wearable device, as detailed by Kapur et al. [1] [2], that allows a user to communicate without any speaking or visible movement. Unlike devices that listen for speech, AlterEgo reads residual neurological signals produced during internal articulation or "silent speech" – often described as one's inner voice, which generally manifests as the slight activation of the internal speech system. The system then uses machine learning to transcribe those signals into text. The resulting system is one that can be used similarly to a voice-controlled virtual assistant, but without the need to actually speak or perform any visible movements or gestures. When paired with a discreet feedback device, such as bone conduction headphones, it gives the experience of having an AI or virtual assistant in one's head.

In this thesis, I present my work on a continuous silent speech recognition system for AlterEgo as part of the research of the Fluid Interfaces Group at the MIT Media Lab. By transcribing residual neurological signals sent from the brain to speech articulators during internal articulation, the system allows one to communicate without the need to speak or perform any visible movements or gestures. It is capable of transcribing continuous silent speech at a rate of over 100 words per minute. The system therefore provides a natural alternative to normal speech at a rate not far below that of conversational speech. This alternative method of communication enables those who cannot speak, such as people with speech or neurological disorders, as well as those in environments not suited for normal speech, to communicate more easily and quickly. In the same capacity, it can serve as a discreet, digital interface that augments the user with information and services without the use of an external device.

I discuss herein the data processing and sequence prediction techniques used, describe the collected datasets, and evaluate various models for achieving such a continuous system, the most promising among them being a deep convolutional neural network (CNN) with connectionist temporal classification (CTC). I also share the results of various feature selection and visualization techniques, an experiment to quantify electrode contribution, and the use of a language model to boost transcription accuracy by leveraging the context provided by transcribing an entire sentence at once.

1.1 Motivation

There are various scenarios in which normal speech is not a viable method of communication. A system capable of transcribing internally articulated words provides an alternative method of communication, whether it be with another person or with a computing device/virtual assistant. In the latter case, the system can serve as a discreet, digital interface that augments the user with information and services without the use of an external device and the disruption from one's environment that often comes with it.

First, the user may not wish to speak out loud for a few reasons. One such reason is that the topic may be personal or not desirable to mention in public. The user may also be in a public setting where others don't want to be disturbed, such as on an airplane or in a library. In situations like these, most people would generally opt for physically interacting with a device, such as their phone, in order to communicate. However, the need to physically interact with a device that could otherwise be used tangentially may serve as a distraction or an inconvenience when trying to complete a task. Second, the setting may not be conducive to a normal conversation with another person or a device. For example, one may be near a large crowd or in a factory, where the surrounding noise makes normal speech difficult to discern. In this case, conversation or use of a voice-controlled virtual assistant is not just undesirable, but infeasible.

Finally, and most importantly, many people have difficulty speaking or are unable to speak entirely (conditions known collectively as dysarthria) due to a variety of speech or neurological disorders. The National Institute on Deafness and Other Communication Disorders [3] states that approximately 7.5 million people in the United States alone (over 2%, by Feb. 2019 UN estimates) have trouble using their voices to communicate. All of the conditions associated with dysarthria affect one's ability to speak, but many do not affect one's ability to internally articulate, the result of residual signals from neuromuscular pathways that remain functional. In fact, several user studies on the use of AlterEgo by ALS and MS patients showed that all participating patients, with varying stages of their respective disorder, were able to produce signals to the same extent as someone without any speech or neurological disorder. Furthermore, in successful user trials, several patients with ALS and MS were able to use our system to accurately transcribe a small vocabulary of silent speech into text, which was then spoken aloud using text-to-speech software. Current systems that allow affected patients to speak, known as speech generating devices, are slow to use as they rely on an unnatural means of input, such as a headmouse or eye gaze. Many are also tiring to use and prevent the user from making eye contact with others while communicating. Therefore, a system that can accurately transcribe sentences from internal articulation, a fairly natural means of input, and do so in real-time, could potentially help millions of people communicate more easily.

1.2 Background and Related Work

Much work has been published on automatic speech recognition systems, and large datasets have enabled rapid advancement in the field. Far less work has been published with a focus on automatic recognition of silent speech, and doing so continuously without any visible movement.

1.2.1 Automatic Speech Recognition

Automatic speech recognition (ASR) systems are systems capable of transcribing speech into written language. The capabilities of ASR systems vary greatly, so one must be more specific when discussing these systems. In their review of speech recognition systems, Saksamudre et al. [4] introduce a classification scheme based on a system's robustness and level of understanding of natural speech. The scheme classifies systems into the following levels, listed by level of understanding from most-limited to most-inclusive: isolated-words, connected-words, continuous speech, and spontaneous speech. The two levels relevant to this work are connected-words and continuous speech systems. Connected-words systems recognize primarily individual words, although they may recognize several words in succession with sufficient pauses between them. Users of these systems must be aware of the constraint placed on them – a pause or delimiter signal of some sort is necessary for accurate transcription of more than one word. This limits their ability to communicate naturally and at a high rate of speech. These models also only take as input a single word at a time, which results in each word being predicted independent of all others. The original AlterEgo system, as described by Kapur et al. [1] [2], was a connected-words system, and therefore was subject to all of the aforementioned problems. Continuous speech systems, on the other hand, eliminate the need for clear divisions between words and are able to transcribe entire sentences or phrases at a time. The system presented in this thesis is categorized as a continuous speech system for this reason. Another benefit of this type of system is that the input is an entire sentence or phrase, so the context provided by surrounding words can be utilized in the prediction process to improve prediction accuracy.

Continuous speech recognition systems for audible speech are already ubiquitous in today's world, and the technology is a major selling point for countless new devices. Speech is a particularly rich input modality, and the technology required to capture audio signals is no barrier to the development of these systems – even simple microphones generally record at incredibly high sampling rates with very little systemic noise. There are also many large datasets that can be used to train a speech recognition system on regular speech. Unfortunately, none of the same can be said for the modality of silent speech, making the development of reliable continuous silent speech recognition systems an active area of research.

1.2.2 Silent Speech

The term "silent speech", as used in this thesis, refers specifically to internal articulation, which is the deliberate but subtle neurological activation of internal speech articulators that is often observed while reading very intently. Importantly, this action does not result in any sound *or visible movement*. In other papers, the term "silent speech", as well as the more loosely-defined terms "subvocalization" and "subvocal speech", are often used to describe actions that result in some visible movement or audible speech. Regardless of the level of movement or sound involved, all types of subvocalization are a natural means of input as derivatives of natural speech. With silent speech, rather than performing a gesture or coded interaction to indicate a specific intent, one only needs to silently speak their intent, as if talking to oneself internally. This is also arguably a more natural means of input than using a physical device like a keyboard or mouse, as with these devices we must learn how to efficiently express our ideas, unlike with natural speech. Therefore, silent speech shares all the benefits of communicating with regular speech, without the need for any overt action by the user.

Research into subvocalization started as early as 1970 when Eriksen et al. [5] and, later in 1973, Klapp et al. [6], explored the concept of "implicit speech" in reading. It was suggested that subvocalization may play a role in subjects' understanding of previously unseen words, as they would often pause and subvocalize new words prior to reading them out loud. For this reason, subvocalization was thought to be a precursor of spoken language, or another representation of the spoken language itself. This, coupled with drastic improvements in speech recognition systems over the years, led researchers to consider the possibility of transcribing subvocal speech into text.

The NASA Ames Research Center began research into subvocal speech recognition systems in 2003. In one paper, Jorgensen et al. [7] describe training a neural network model to recognize 6 to 10 subvocal words, including digits zero through nine. Their system used sub-audible (or sotto voce) speech and relied on electromyographic (EMG) signals from visible muscle movement – a clear distinction from our system. Using 100 training samples of each word, they achieved an accuracy of 92% on recorded test samples. Their conclusion upon experimenting with larger vocabularies is that a 20-word recognition task seems feasible, but recognition of a significantly larger vocabulary is not. As such, they suggest that the system be used with a small vocabulary specialized to the task at hand. They also mention that the system still suffers from being specific to a single user and sensitive to noise, electrode locations, and physiological changes in the user. These issues have been found to affect the AlterEgo system as well.

Several other research groups have experimented with subvocal speech recognition systems. There has been significant exploration in using invasive systems with direct brain implants to recover signals generated during subvocal speech. Brumberg et al. [8] report that the use of intracortical microelectrode arrays shows promising results in early-stage trials with the goal of providing a speech prosthetic. Similarly, in a recent Nature article, Anumanchipalli et al. [9] discuss their method of synthesizing intelligible speech from high-density electrocorticography (ECoG) signals. Using ECoG data collected from patients while audibly speaking sentences at a normal conversational rate, they are able to decode the signals into an intermediate audio spectrogram, and then into sythesized speech. Sentence-level intelligibility tests, which asked subjects to listen to and transcribe the synthesized speech, reported word error rates of 31% for a 25-word vocabulary, and 53% with a 50-word vocabulary. These results were for fully-audible speech, while they report that results for mimed speech (still with significant visible movement) were significantly worse.

There has also been significant exploration of non-invasive systems, with the hope of scaling beyond clinical use. Wand et al. [10] use a non-invasive system to transcribe EMG signals from subvocal speech into text, achieving a fairly low word error rate on a large vocabulary, but with clearly visible movements of the mouth. This work is continued by Jou et al. [11], who evaluate a continuous subvocal speech recognition system, but still with significant movements and a high word error rate.

All known studies, including those mentioned here, deviate from the goals of AlterEgo in a significant way. The vast majority focus only on transcribing individual words with a connected-words level system or continuous speech with a low transcription rate. As previously mentioned, the drawback of this, other than taking significantly longer to communicate the same amount of information, is that the system requires a significant pause between words and therefore feels much less natural to use. The primary goal of our system, however, is to transcribe continuous silent speech with a relatively high transcription rate. The few studies that do use continuous recognition systems do not focus on transcribing internal articulation specifically. Instead, their focus is on transcribing subvocal speech with significant movement of mouth or facial muscles, or with significant sound produced. While this is acceptable in many situations, our use cases require little to no visible movement or sound, especially if our system is to aid patients who are unable to exert this level of muscular control.

1.2.3 Sequence-to-Sequence Models

Continuous speech recognition systems make use of sequence-to-sequence models, also known as sequence prediction or sequence generation models. These models produce a variable-length sequence as output, as opposed to the fixed length output produced by classification models. Continuous speech recognition requires the use of these models because the input to the system is a signal representing an entire sentence, and the output is a variable-length sequence of words which form the full transcribed sentence. Other examples of tasks that require sequence-to-sequence models are machine translation, handwriting recognition, image captioning, and speech generation.

The learning process for sequence-to-sequence models differs significantly from that of classification models. Using continuous speech recognition as an example, not only must the model learn to classify portions of a signal as the correct word, it must also learn how many words are present in the input signal and which portions of the input signal correspond to each word in the output sequence. This surjective function of input to output that must be learned by the model is known as the alignment. For some applications, the ordering of the input and output segments may not be the same, as is the case with machine translation due to grammatical differences between languages. Fortunately, this is not the case for speech/silent speech recognition systems, though, because the words in a sentence are always said in the same order that they are written. This simple fact provides greater flexibility in choosing a sequence-to-sequence modeling approach, which will be discussed in more detail in Section 2.3.

The traditional approach to label sequence prediction, before heavy use of the now-common recurrent neural network (RNN) encoder-decoder models, was to use a hybrid neural network and hidden Markov model (HMM) model. The neural network would be used for classification at each timestep of the input sequence, and the HMM would decode these predictions into the output label sequence, as discussed by Juang et al. [12] in the context of speech recognition with HMMs. However, the need to train both the classifier and decoder separately is undesirable. Currently, the most commonly used approach for sequence prediction is the use of RNN encoder-decoder models. As explained by Lu et al. [13], the approach uses an encoder RNN model to transform the input sequence into a fixed-length vector representation, and a decoder RNN model to transform the fixed-length vector, which theoretically contains information on the entire input sequence, into an output sequence of labels. The loss is calculated from the output of the decoder model, but because the decoder takes as input the output of the encoder model, the two are trained simultaneously during loss optimization. There are, however, a few downsides to using this approach. For long-term dependencies to be reliably modeled, one must use a long short-term memory (LSTM) or gated recurrent unit (GRU) model, both of which are slow to train. A simple RNN with a less complicated activation function takes far less time to train than an LSTM or GRU, but due to the issue of exploding/vanishing gradients [14], cannot manage long-term dependencies. Also, RNNs in general are prone to overfitting, resulting in poor generalization. The approach primarily used in this work, known as connectionist temporal classification, is capable of overcoming both of these obstacles because of its simplicity and ability to be used with either an RNN or CNN (see Sections 2.3.3 and 2.3.4).

Chapter 2

Approaches

This chapter will describe the signals of interest and how they are captured, the signal processing techniques used to separate meaningful features from noise, several methods of feature selection and visualization, and the machine learning approaches used to transcribe these signals into a sequence of words. Additionally, we detail the integration of a language model for increased sequence prediction accuracy.

2.1 Signal Capture and Processing

2.1.1 Signal Capture

In creating a subvocal speech recognition system, the signals of interest are the individual neurological activations of internal speech articulators. As a non-invasive device, there is no way for AlterEgo to directly record these signals. Instead, they must be collected externally, far from the actual source. The best method for doing this is with electrodes that can pick up on subtle disruptions in the electric field surrounding speech articulators. With these electrodes placed on the face and neck regions, as in Figure 2-1, an aggregation of the desired neuromuscular signals is apparent during internal articulation. The device uses 8 signal electrodes and reference and bias electrodes, measuring the potential difference between each signal electrode and the reference electrode (with the bias electrode used for on-device data process-

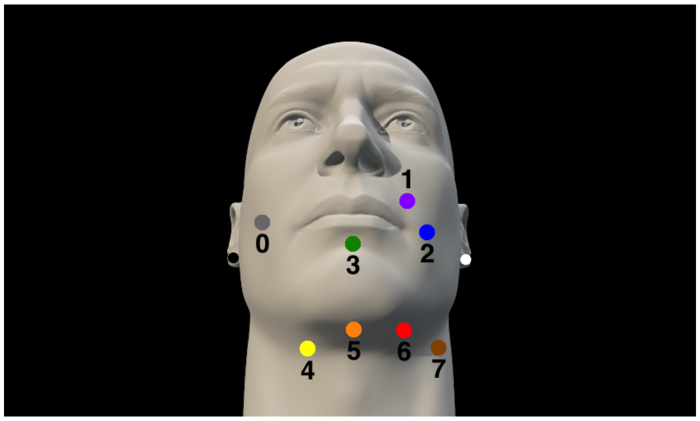


Figure 2-1: A typical arrangement of 8 signal electrodes (colors and gray, on face and neck) and reference and bias electrodes (white and black, interchangeable on earlobes).

ing). Figure 2-1 shows a typical arrangement of the electrodes. This arrangement, or at times a subset of the electrodes, is used in all experiments detailed in this paper. Many subsets of these 8 signal electrodes have been used without a noticeable drop in accuracy, as some of the electrodes capture similar signals due to proximity to the same muscles. For example, all electrodes placed on the neck (electrodes 4-7 in Figure 2-1), capture highly correlated signals, and therefore the system generally performs the same with just one or two of them.

This method of using surface electrodes to capture signals from internal articulation also captures a large amount of noise introduced by the body, as would be expected when collecting distant physiological signals. The data collection device also introduces a secondary source of noise, as well as a slight baseline drift in the signals over time. This noise proves troublesome because the signals collected via this method, as shown in Figure 2-2, are already comparatively weak (generally in the range of single digit to low double digit microvolts), resulting in data with a low peak signal-to-noise ratio. Therefore, some signal processing is required to extract the features of interest before a model can learn from the data.

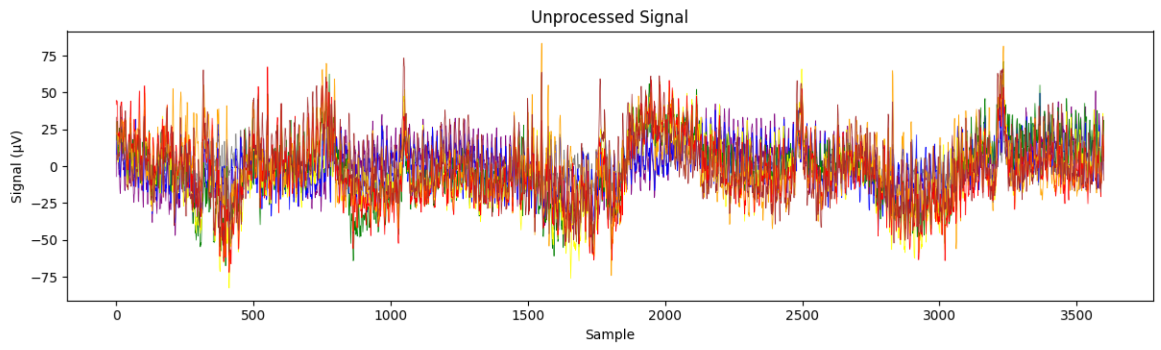


Figure 2-2: The unprocessed signal of a sequence of three words ("tired", "hot", "i"). The 8 data channels are colored according to the electrode placement shown in Figure 2-1.

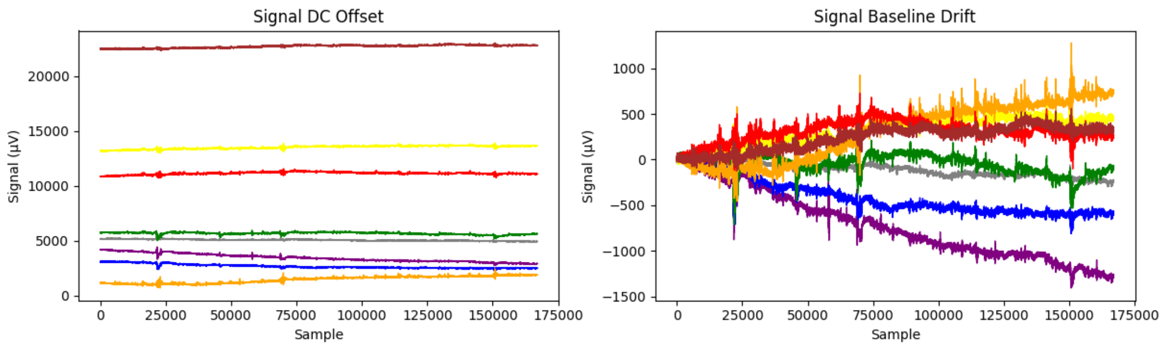


Figure 2-3: The DC offset (left) and baseline drift (right) of an unprocessed signal over about 3 minutes. The signals in the right plot are normalized by their initial value to show the baseline drift, but in reality are the same as in the left plot.

2.1.2 Correcting for DC Offset and Baseline Drift

The variation in electrode placement relative to the reference electrode creates a large DC offset, which manifests as a seemingly arbitrary offset in each data channel. Additionally, the device used to record potential differences between signal and reference electrodes introduces a very low frequency source of noise. While this noise is low enough in frequency that it affects only the baseline, or long-term mean amplitude of a signal, this drift is rapid enough that the baseline can drift by up to a few hundred microvolts (an order of magnitude larger than the amplitude of the signal of interest) in less than a minute. These two effects are clear in signals collected over a long period of time, as shown in Figure 2-3. They are also apparent on shorter time scales, albeit much more subtly. The rate and direction of baseline drift is unpredictable and the

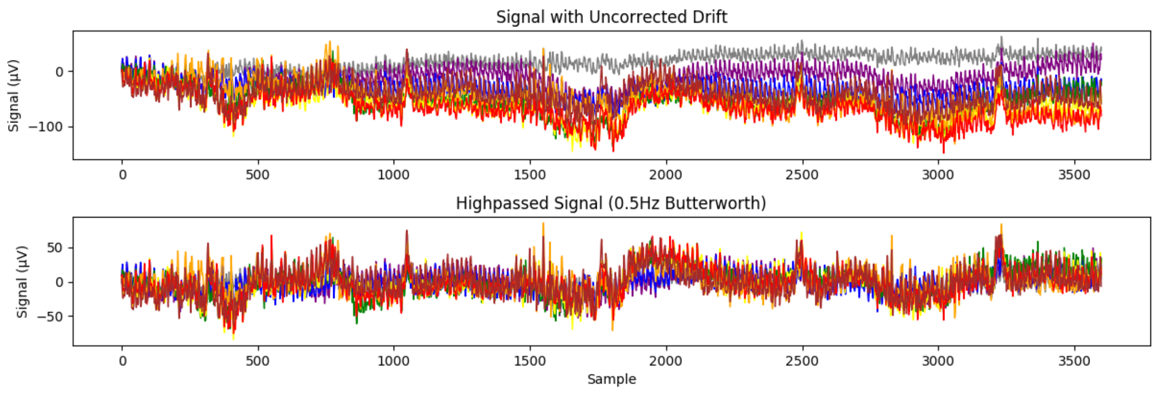


Figure 2-4: A Butterworth highpass filter is used to remove frequencies less than 0.5 Hz. The uncorrected signal (top) baseline drifts significantly over a period of only about 3.5 seconds, whereas the highpassed signal (bottom) does not.

baseline of each channel changes independently of the others. This negatively affects a model's ability to recognize features within a signal, as the relative amplitudes of channels change in a way that is difficult for a model to learn.

To remove these artifacts, the signals are normalized by their initial values and frequencies lower than 0.5 Hz are highpassed out using a 1st-order Butterworth filter. Initial value normalization prior to the filtering is necessary to avoid severe artifacts upon applying the Butterworth filter. Afterwards, the signals are normalized such that the mean amplitude is zero, in order to center the signals. Figure 2-4 shows the drift correction for a single recording of about 3.5 seconds in length. With the correction, one can see the correlation of channel amplitudes across channels, which is less obvious without the removal of the baseline drift.

2.1.3 Noise Removal Techniques

There are three sources of noise present in our data, after the DC drift has been corrected. The first is a 60 Hz power line hum resulting from interference from the power source. The second is a heartbeat artifact, which features prominently in the data regardless of electrode placement. The third is noise originating from the body due to the distance between the signal source and the surface, where signals are recorded.

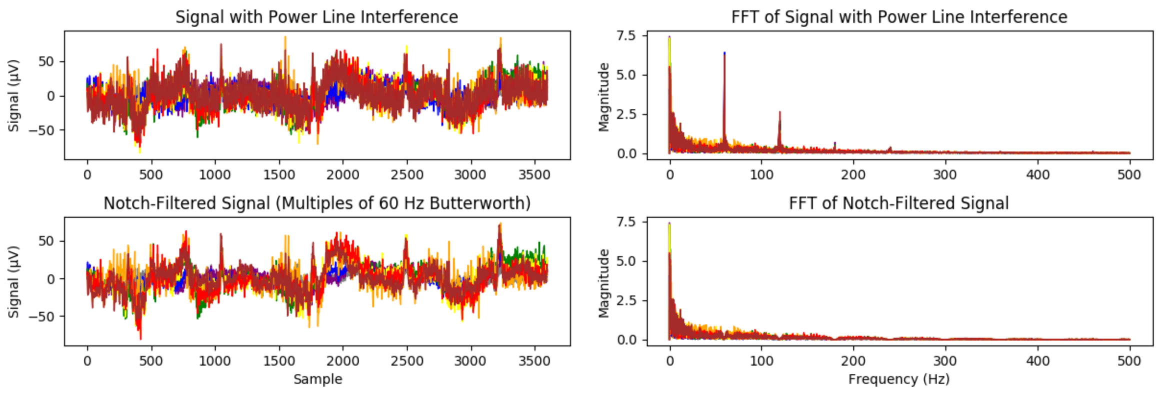


Figure 2-5: Several Butterworth notch filters, with bands at multiples of 60 Hz, are used to remove the power line hum. The notch-filtered signal and its frequency domain (bottom) show almost no noise at multiples of 60 Hz, unlike the signal with power line interference (top).

The simplest, yet still effective, approach of the many approaches documented by Mewett et al. [15] for removing the power line hum is by using a notch filter – a band-stop filter that removes a narrow band of frequencies. For best results, notch filters are applied not just to the 60 Hz frequency, but to all multiples of 60 Hz less than the sampling rate. Looking at the frequency domain of the signal, all multiples of 60 Hz have a significantly higher magnitude than the surrounding frequencies, so removing solely the 60 Hz band does not remove all of the power line hum. Figure 2-5 shows the signals and frequency domains before and after these filters are applied.

Heartbeat artifacts are very prominent in the collected data and cannot be controlled for, unlike artifacts from actions such as blinking or head movement. Fortunately, heartbeat artifacts are also very distinct. However, they are difficult to remove while preserving all other features, because they resemble narrow features of interest and occur at intervals of relatively low frequency. As such, they are not entirely removed by the other noise reduction techniques discussed in this section, which focus on removing high frequency noise while preserving lower frequency features of the data. Therefore, rather than using a band-stop filter, a wavelet convolution is applied to select for regions that resemble a heartbeat artifact. More specifically, a Ricker wavelet is used as an approximation of a heartbeat artifact. The convolution is then subtracted from the original signal to reduce the prominence of heartbeat arti-

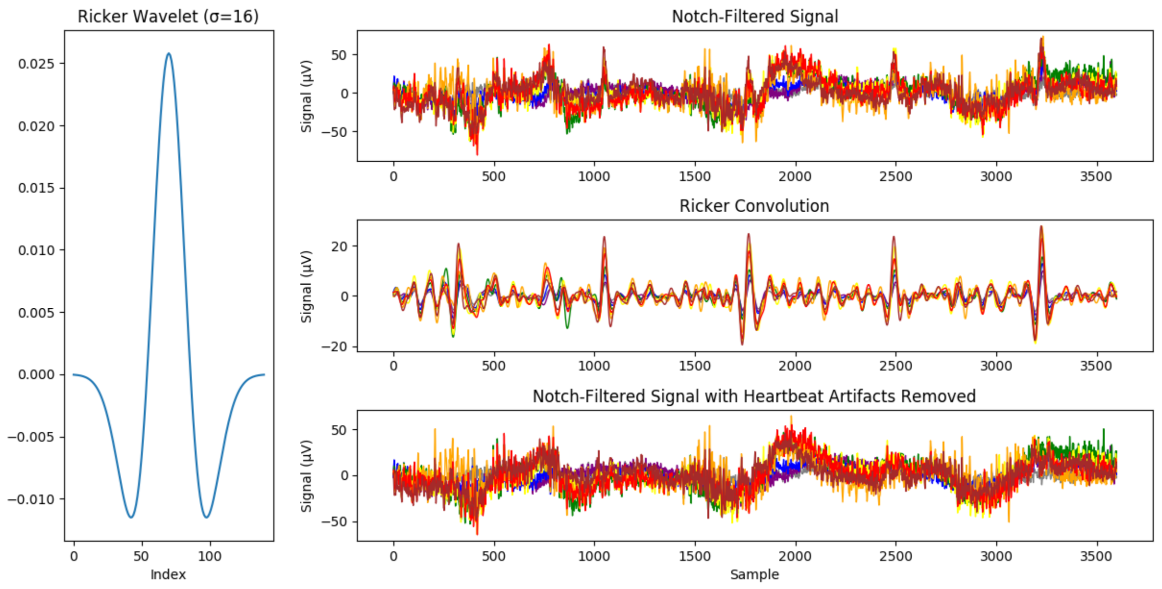


Figure 2-6: A Ricker wavelet (left) is convolved with a signal with heartbeat artifacts (top). The convolution (middle) is then subtracted from the signal, resulting in approximate removal of heartbeat artifacts without significantly distorting the original signal (bottom).

facts without significantly distorting the original signal. Figure 2-6 shows the Ricker wavelet, an example convolution, and the resulting signal with reduced heartbeat artifacts.

Finally, the body introduces a lot of high frequency noise that is of little to no use in our sequence prediction task, as the features we are trying to learn are much lower frequency. The solution is to simply apply a bandpass filter to the range of frequencies we are interested in, which we have determined through analysis of the frequency domain to be 0.5 to 8 Hz. Since filters only reduce the magnitude of the target frequencies, often times some high frequency noise remains. An alternative solution, if one wishes to ensure complete removal of all frequencies outside the desired band, is to calculate the FFT of the signal, zero out the frequencies directly, and then use the inverse FFT of this modified frequency domain. Both approaches result in similar accuracies when tested with a sequence prediction task. With this filter applied, we arrive at our fully processed signal, as shown in Figure 2-7.

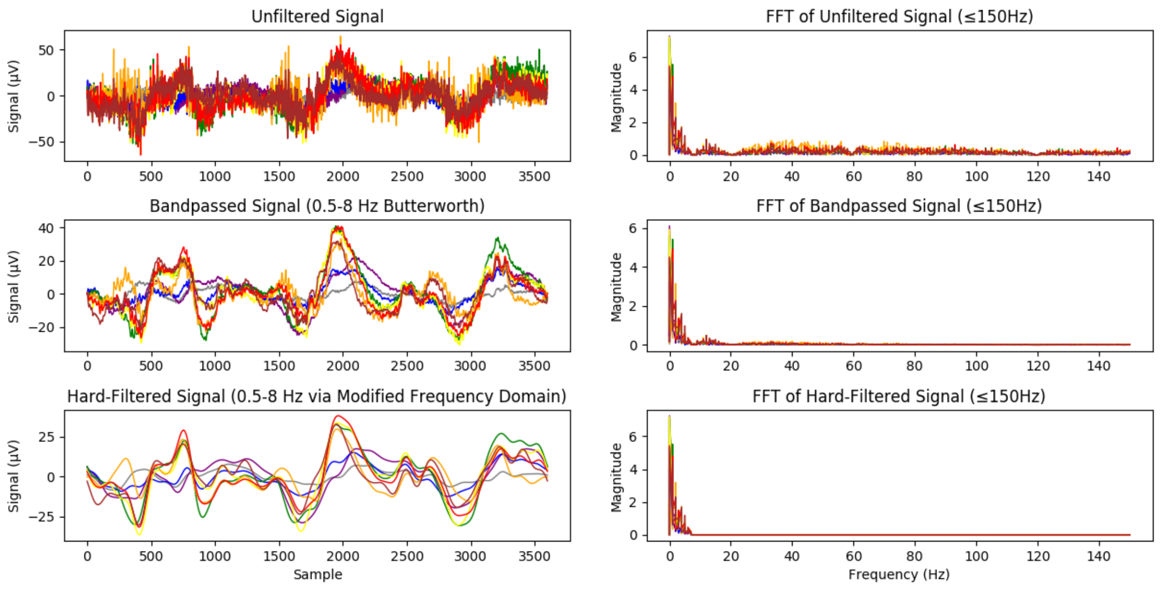


Figure 2-7: A Butterworth bandpass filter is applied to a signal (top) to remove most noise outside the frequency band 0.5-8 Hz, resulting in clearly visible low-frequency features (middle). Alternatively, modifying the frequency domain removes completely all frequencies outside of this band (bottom).

2.2 Feature Selection and Visualization

We also used various feature selection techniques in an attempt to emphasize the meaningful features of a signal, while also providing a better way to visualize the data. We experiment with wavelet convolutions and methods of unsupervised learning, namely k-means clustering to group similar features and an autoencoder to transform the data into a more-meaningful, lower-dimensional latent space, while at the same time removing uncommon features and noise.

2.2.1 Wavelet Convolutions

Wavelets can be used to extract specific features from a signal, and are generally used to select for areas of a signal that are closely correlated with the wavelet. For example, a sine wavelet of a certain frequency will result in a convolution that emphasizes the regions of a signal with that particular frequency. Therefore, if a particular feature of the signal is difficult for a neural network to learn, one can input the wavelet-convolved signal to provide the network with the pre-selected feature. This transformation

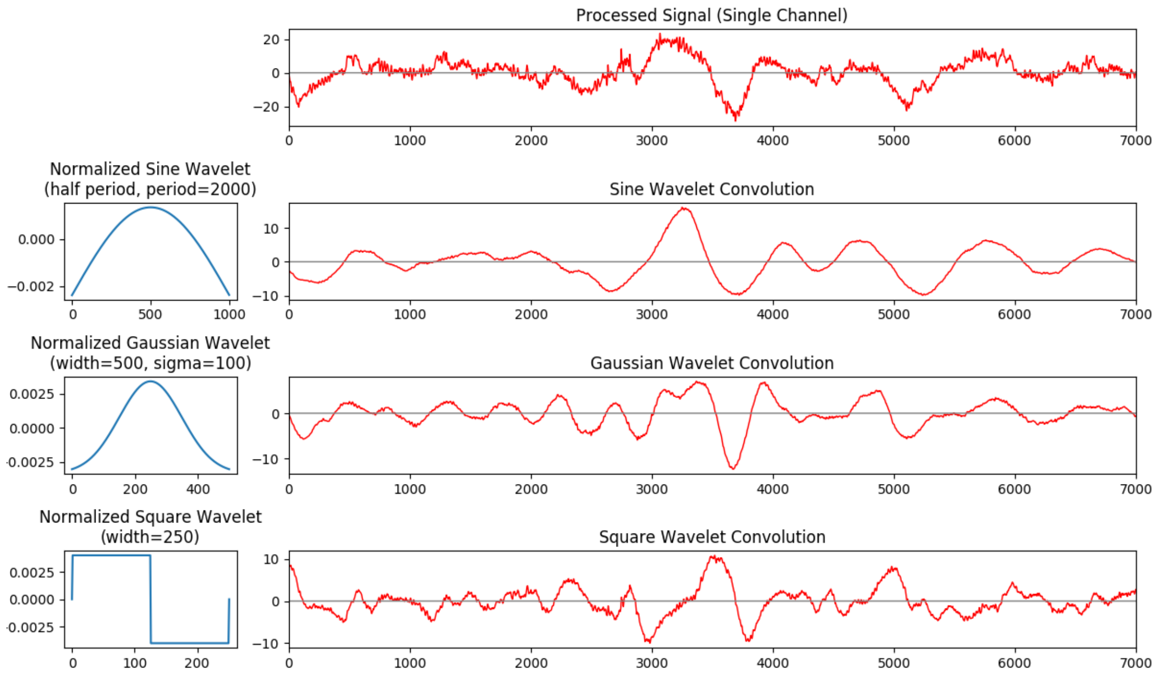


Figure 2-8: Examples of normalized sine, Gaussian, and square wavelets, all centered around zero, and the corresponding convolutions on an example signal.

allows the network to learn from the features of interest without having to learn to recognize them in the raw signal.

The collected neuromuscular signals generally contain numerous peaks and falls of varying amplitudes, centered around zero. Therefore, the most revealing wavelets tend to be those correlated with different types of peaks and falls centered around zero. Sine wavelets of different frequencies emphasize peaks of those same frequencies very well, as they essentially filter out other frequencies. Gaussian wavelets perform similarly, but without eliminating so much of the features at different frequencies. We also experimented with wavelets such as square wavelets, which do not resemble a sinusoidal wave and therefore do not have as much of a smoothing or denoising effect, while still serving to emphasize similar features. Figure 2-8 shows these three types of wavelets and the resulting convolutions on an example neuromuscular signal.

Wavelets provide a way to select a subset of the features present in a signal by choosing a specific wavelet, and are therefore also useful in visualizing different features of a signal. However, we have found that our neural network models did not perform significantly better when wavelets were applied, even if the resulting convolutions were used simply to augment the raw signal. This indicates that the model has no issue learning the full set of features present in the data. With that said, for specific vocabularies, we have seen a marginal increase in classification accuracy using sine wavelets, presumably because they selected the most information-rich frequencies and filtered out the rest.

2.2.2 K-Means Clustering

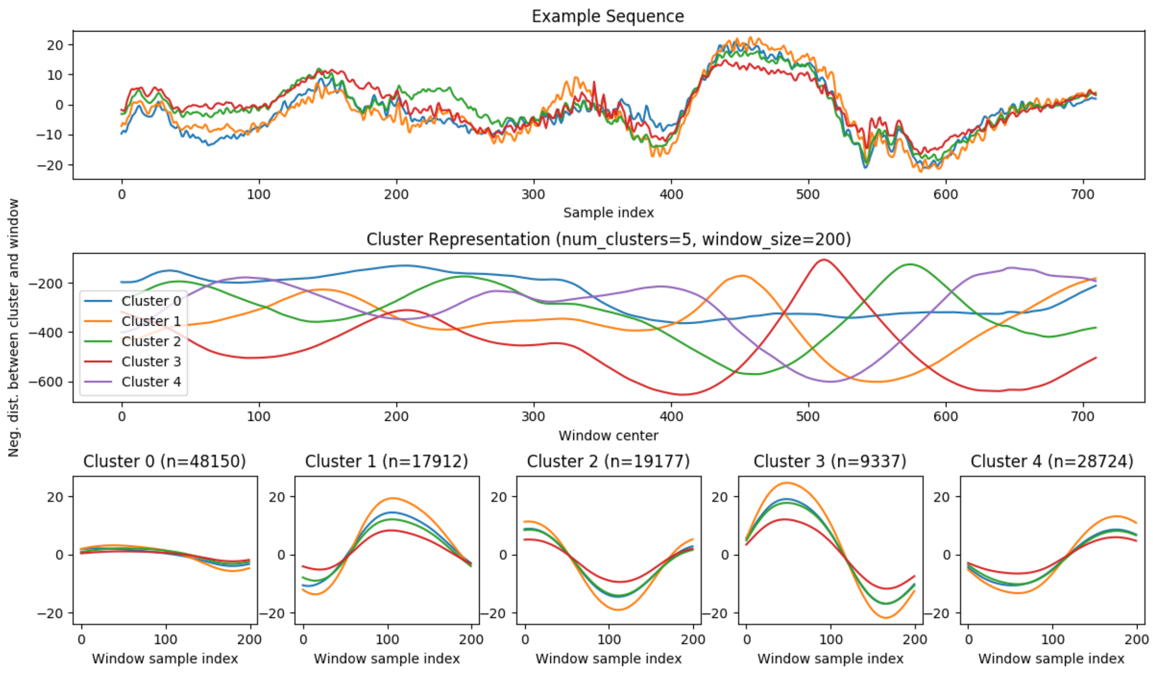


Figure 2-9: An example sequence (top), the clusters chosen from a large dataset of similar sequences (bottom), and the cluster representation of the example sequence (middle).

Unsupervised learning can be used to find features in data without knowing what distinguishing features are present. K-means clustering is one way to select for and visualize features of the signals, and can also serve as a form of dimensionality reduction. We use fixed-size windows of many example signals to determine the k clusters, which represent the k most distinct features found in the data. One can then transform the signal into the cluster space by calculating the distance from each cluster mean at each timestep. In essence, this resulting cluster representation is a sequence where each channel corresponds to the likelihood of one of the k most likely features (as determined by the clusters).

Figure 2-9 shows an example sequence, the 5 clusters chosen from a large dataset of similar sequences, and the cluster representation of the example sequence. A benefit of this approach is that it finds a good representation of the k most distinct features found in the data, and therefore each channel of the resulting cluster representation selects for a unique feature. This is also beneficial in that no channel is highly correlated with any other channel, as can clearly be seen in the middle plot of Figure 2-9, which helps to maximize the amount of information contained in each channel while also providing a good visualization. For this reason, there is no need to manually design filters to select features of interest, as in Section 2.2.1.

2.2.3 Autoencoder and PCA of the Latent Space

An autoencoder is another method of unsupervised learning that can be used to select and visualize significant features, as well as remove uncommon features and noise from the signal. An autoencoder works by learning how to encode data into a more-meaningful, lower-dimensional latent space by reducing the amount of information the sequences can be represented in. At the same time, it learns how to decode this encoding, or condensed representation, into a reconstructed signal. Reducing the amount of information the signal can be represented in forces only the most significant/common features to be learned, leading to the removal of uncommon features and noise.

We used a CNN to encode our data, similar to the CNN described in Section 2.3.4. It uses three fully connected layers as a decoder, the first of which is only 32 units – this represents the dimensionality of the latent space that we are transforming our data into. It operates on a fixed-size window and is trained on windows from all of the recordings of one of the datasets we collected. Figure 2-10 shows example output from an autoencoder with a window size of 1000 samples. One can see that signals reconstructed from the autoencoder's condensed representation are devoid of smaller,

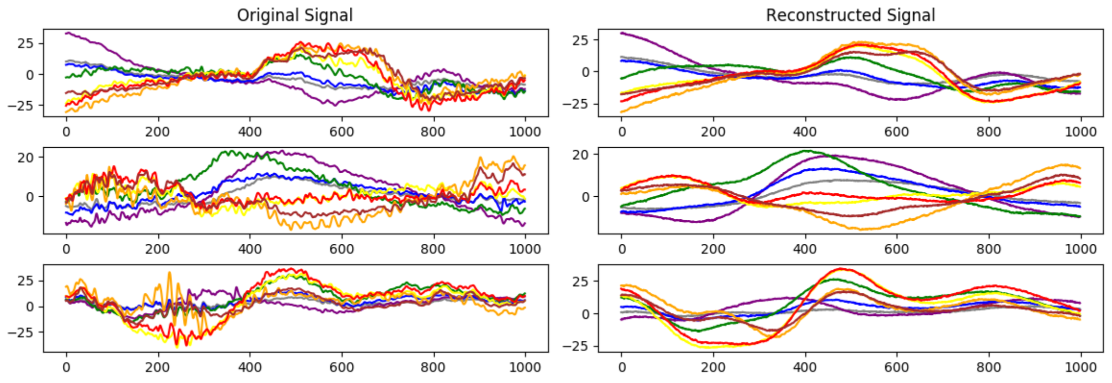


Figure 2-10: Example output from an autoencoder with a window size of 1000 samples. One can see that signals reconstructed from the autoencoder's condensed representation are devoid of smaller, high-frequency features, which the autoencoder cannot afford to encode in only 32 units of information. It also removes less common features, like periods of extreme noise that occur very rarely.

high-frequency features, which the autoencoder cannot afford to encode in the only 32 units of information. It also removes less common features, like periods of extreme noise that occur very rarely. We chose to use 32 units as the size of the condensed representation because it results in the desired level of detail. Fewer units results in less detail, and significantly more units allows more of the higher frequency features and noise to be learned.

When we tested our sequence predictions models, as described in Section 2.3, with condensed representations of the signals, rather than the actual signals, we received similar results (albeit with much faster training). This indicates that the autoencoder is able to learn the same features as the CNN and LSTM models used in sequence prediction. For this reason, we use the autoencoder for visualization, but not directly in our sequence prediction models.

Once a latent space that accurately represents the signals is learned, we can then attempt to directly visualize what features the autoencoder has learned. One way to perform this visualization is to manually modify each unit of the encoding and see how it affects the reconstructed signal (i.e. the decoded encoding). However, the latent space dimensions are not guaranteed, and are in fact very unlikely, to be independent of each other, making it difficult to visualize the features each unit affects.

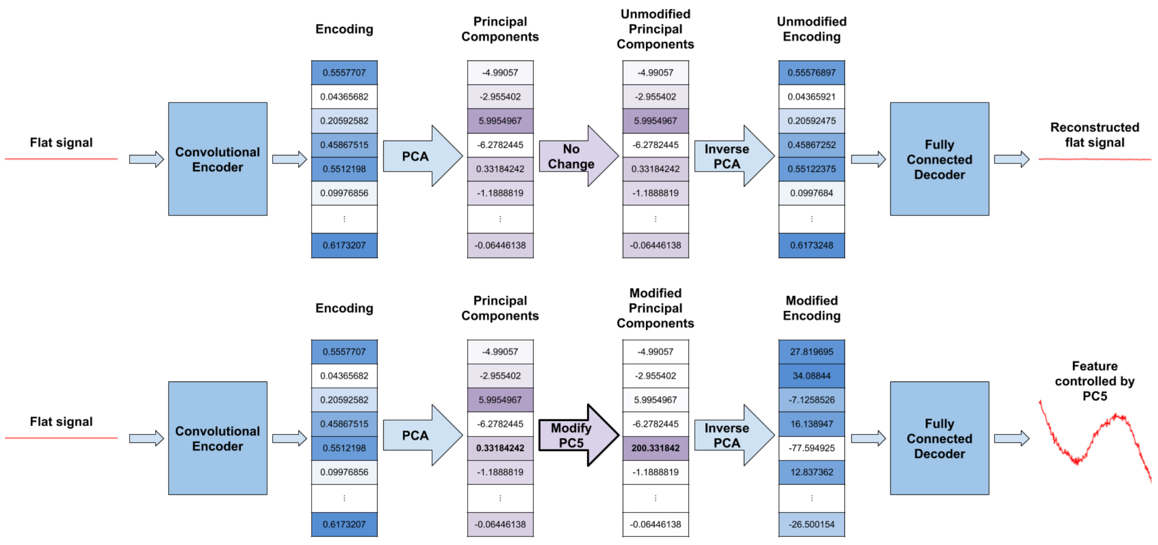


Figure 2-11: The changes to the reconstructed signal upon modifying one of the principal components reveals what features have been learned by the autoencoder that are controlled by that component. The process without any changes to the principal components is shown on top, and the results upon modifying PC5 are shown on bottom.

A solution to this is to use principal component analysis (PCA) to find an orthogonal transformation that transforms the data into a space with linearly uncorrelated dimensions. In this transformed space, each component will control a feature (or set of features) in the reconstructed signal independent of all other components. After specific principal components have been modified, we can use inverse PCA to transform the principal components back to the latent space encoding and run it through the decoder to see the reconstructed signal. The changes to the reconstructed signal upon modifying one of the principal components reveals what features have been learned by the autoencoder that are controlled by that component. Figure 2-11 demonstrates this process. Doing this for all principle components is one way of visualizing what features an autoencoder has learned. We trained an autoencoder with a latent size of 32 on a large dataset and performed this analysis on all 32 of the principal components. The results are shown in Figure 2-12. Also shown in the figure is the explained variance, or eigenvalue, of each of the principal components, which gives a measure of how prevalent the features controlled by each component are. One can see that the first 6 or 7 principal components control very distinct features, whereas the

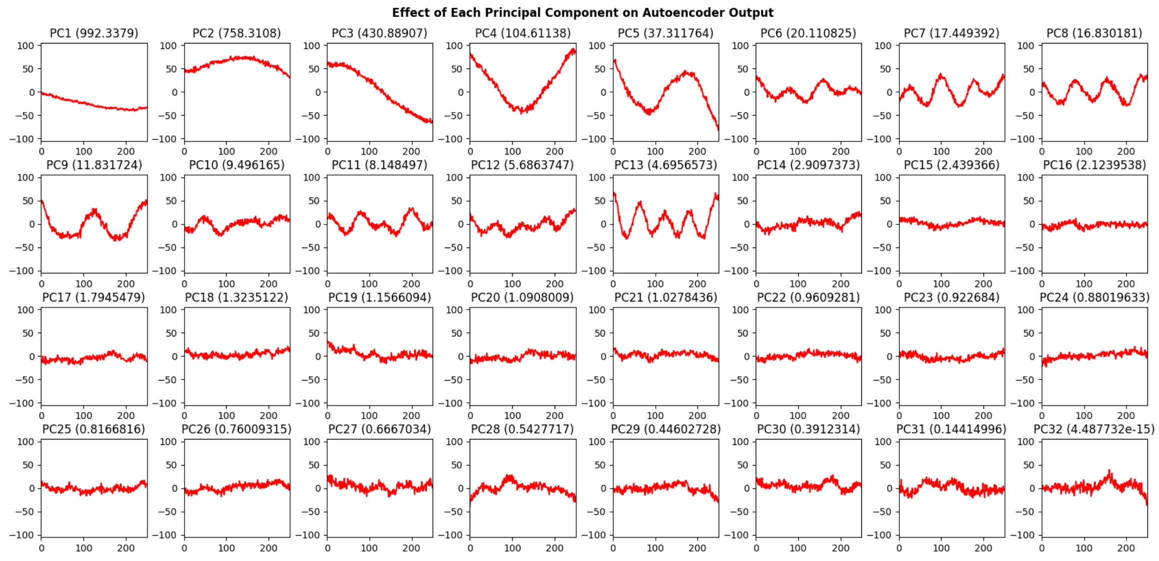


Figure 2-12: The effect of each principal component on the reconstructed signal, using an autoencoder with latent size 32. Visualizing the effects is done by manually modifying the principal components in the transformed encoding space and seeing the change in the reconstructed signal. Also shown is the explained variance, or eigenvalue, of each of the principal components, which gives a measure of how prevalent the features controlled by each component are. One can see that the first 6 or 7 principal components control very distinct features, whereas the higher principal components seem to have features in common, or show no significant features at all. This is reflective of the features actually seen in the dataset.

higher principal components seem to have features in common, or show no significant features at all. This is reflective of the features actually seen in the dataset.

2.3 Sequence Prediction

Many sequence prediction approaches are suitable for transcribing neuromuscular signals from internal articulation into text. The primary approach discussed in this paper uses connectionist temporal classification [16]. However, we will share a naive approach to sequence prediction through the use of classification models. The challenges encountered in this approach highlight the difficulties in using classification models for sequence prediction, and the benefits of sequence-to-sequence models.

2.3.1 Dual Word and Activation Classifier

If one is willing to accept a few constraints on the use of the system, a simple approach to sequence prediction is to use two classification models. The first model, the word classifier, must take as input a fixed-size window of the signal and predict which single word it represents. The second model, the activation classifier, must also take a fixed-size window, but instead predicts whether the signal represents activity or non-activity (i.e. whether the user was silently speaking at the moment or not). In this approach, the activation classifier plays the important role of determining the boundaries between words. When it recognizes a period of activity, the word classifier will then output its prediction for that window of the signal. Together, these independently-predicted words form a sentence.

While neither of the models discussed in this dual-classifier approach are sequence-to-sequence models, used together they can achieve something along the same lines with reasonable accuracy. However, there are still several fairly large negatives to this approach. First, both models operate on a fixed-size window (although the models need not share the same window size). This requires that each word of the vocabulary can be accurately represented in a window of that size, which is problematic for very short or very long duration words. Also, having to choose an appropriate window size for these models introduces another hyperparameter that must be tuned in order to get accurate results. Second, accurate transcription of a series of words relies on high accuracy of both models. If the activation classifier incorrectly predicts the user's intent, then the result will be incorrect regardless of the word classifier's accuracy. Finally, and most importantly, this approach imposes an additional constraint on the user of the system. They must include long enough pauses between words for the activation classifier to recognize a period of non-activity, preventing one from articulating at a natural rate.

A similar approach is to use a single classifier that classifies a portion of the signal as any one of the words or as non-activity. This alternative approach is used similarly to the one described above, except that the two classifiers are combined into one.

While this method simplifies the prediction process by referencing only one model, it has proven to be significantly more error prone for larger vocabularies.

For both the window and activation classifier we use a deep CNN consisting of five convolutional layers (with filters counts [64, 128, 256, 512, 512] and filter sizes [12, 6, 3, 3, 3], respectively) each with a max pooling of size and stride 2, dropout with rate 0.4, one fully connected layer of size 256 units, and a softmax output layer. All layers use a rectified linear unit (ReLU) activation function. The word classifier takes as input a window of size 450 samples (with a sample rate of 250 Hz, this equates to just under 2 seconds of data). The activation classifier, on the other hand, takes as input a much smaller window of size 150 samples (0.6 seconds of data). Additionally, in decoding the activation classifier, we use a window stride of 1/8 of that (about 18 samples, or 75 ms of data). Long periods of activity, represented by a large number of consecutive windows being classified as activity with high-confidence, surrounded by periods of silence, are used to calculate roughly the bounds of each word. The sequence window centered on these bounds is then classified by the word classifier, resulting in a single-label prediction. Over the entire sequence, this dual-classifier process is used to output an entire sequence of labels.

2.3.2 Connectionist Temporal Classification (CTC)

In their 2006 paper, Graves et al. [16] present a method of predicting a sequence of labels for unsegmented sequence data called connectionist temporal classification (CTC). As Figure 2-13 illustrates, a model using CTC is structured and used very similarly to a simple classification model. In fact, the only difference between the two approaches, other than the introduction of an additional class to the classification model, is the use of CTC loss instead of softmax cross entropy loss during training, and the addition of the CTC decoder during prediction (both of which are external to the model).

Before discussing the details of CTC loss and the CTC decoding process, we will share some of the benefits of the use of CTC. The approach is unique in that it allows a classification model – one that outputs a single label prediction – to be used

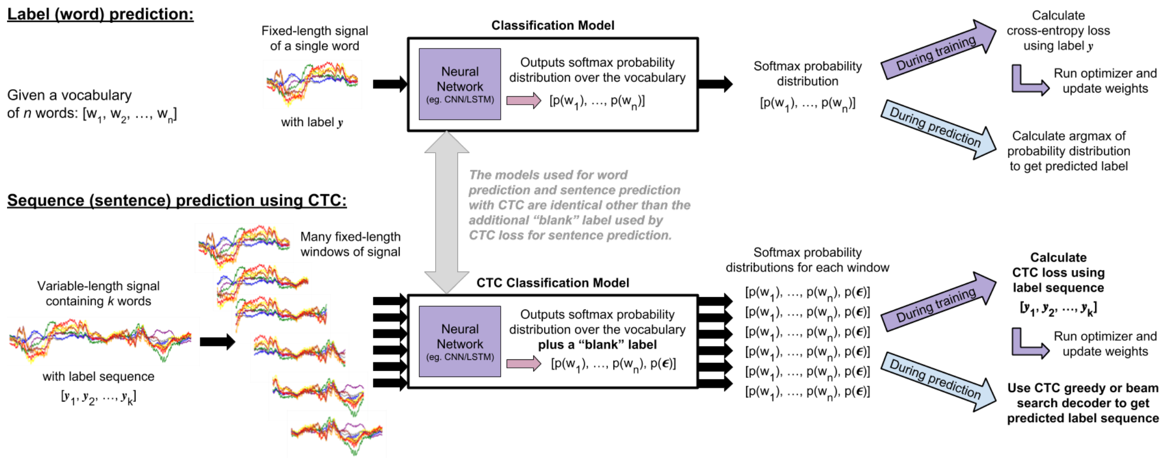

Label (word) prediction:
Given a vocabulary of n words: [w1, w2, ..., wn]
Fixed-length signal of a single word
with label y
Classification Model
Neural Network (eg. CNN/LSTM)
Outputs softmax probability distribution over the vocabulary
[p(w1), ..., p(wn)]
Softmax probability distribution
[p(w1), ..., p(wn)]
During training
Calculate cross-entropy loss using label y
Run optimizer and update weights
During prediction
Calculate argmax of probability distribution to get predicted label
The models used for word prediction and sentence prediction with CTC are identical other than the additional "blank" label used by CTC loss for sentence prediction.
Sequence (sentence) prediction using CTC:
Variable-length signal containing k words
with label sequence [y1, y2, ..., yk]
Many fixed-length windows of signal
CTC Classification Model
Neural Network (eg. CNN/LSTM)
Outputs softmax probability distribution over the vocabulary plus a "blank" label
[p(w1), ..., p(wn), p(ϵ)]
Softmax probability distributions for each window
[p(w1), ..., p(wn), p(ϵ)]
During training
Calculate CTC loss using label sequence [y1, y2, ..., yk]
Run optimizer and update weights
During prediction
Use CTC greedy or beam search decoder to get predicted label sequence

Figure 2-13: A comparison of the single-word prediction and sentence prediction pipelines, showing the minimal difference in structure and function of the two architectures.

directly for label sequence prediction simply by adding an extra class to the classifier's softmax output. All other sequence prediction approaches use a specific architecture designed for sequence generation, such as a recurrent neural network (RNN) encoder-decoder model. CTC's similarity to classification models allows one to easily take the structure of a well-performing label classifier and use it to predict an entire sequence of labels with very few changes to the architecture.

Most importantly, though, knowledge of the label boundaries in the input data (i.e. the alignment between the input and label sequence) is not required during training. In our case, this means that a model using CTC for silent speech transcription is able to train on examples of sentences and their corresponding label sequences without knowing the boundaries of the words within the input data; it need only take an entire unsegmented sentence as input. A sequence prediction model being able to operate on unsegmented data is, of course, expected during prediction, but being able to *train* the model with unsegmented data drastically simplifies the data collection process, and is therefore a significant advantage over other sequence prediction models.

CTC works primarily by focusing on what the paper refers to as temporal classification, in contrast to framewise classification. Framewise classification refers to outputting the most likely label at each frame in the time series data. Temporal classification refers to outputting one label for each subsequence of frames that represents a particular label, and outputting an empty label at intermediate frames. This critical difference makes decoding of a temporal classifier a much simpler task. As the authors explain, a framewise classifier produces a probability distribution over the labels for each frame independent of all other frames. While this is useful for determining the likelihood of a single frame representing each of the labels, this leaves the difficult task of decoding the output of a framewise classifier into the desired label sequence, as the label sequence is necessarily no longer, and generally much shorter, than the number of frames of input data. Traditionally, hidden Markov models (HMMs) have been used for this task, as detailed by Bourlard et al. [17]. A temporal classifier differs in that it produces a probability distribution over the set of labels plus an additional "blank" label at each frame. Using CTC, a temporal classifier is trained in such a way that this "blank" label clearly indicates a boundary between portions of the input sequence that correspond to a single output label. This greatly simplifies the decoding process to the point where an additional model, such as an HMM, is not required. In fact, the greedy decoding process of the CTC output is trivial – simply take the maximum likelihood label at each frame, and then remove blanks and consecutive duplicates.

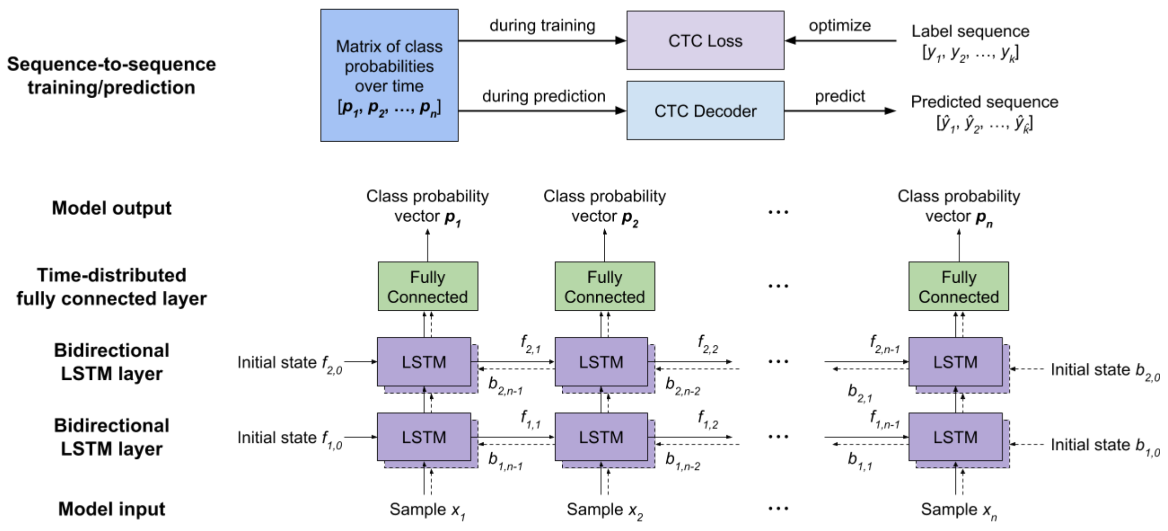
Because the blank label is trained to act as a delimiter between portions of the input sequence corresponding to a single label, this greedy decoding process generally results in a good approximation of the most probable label sequence. However, it is not guaranteed to produce the most probable labeling. For this, one must perform a complete search through the probability distributions across all frames and find the maximum-likelihood path. Using a beam-search decoding process, which makes the decoding more computationally feasible than a complete search, the decoding is guaranteed to find the most probable labeling (given a wide enough beam-search). This beam-search CTC decoding process is used for all experiments discussed in this thesis. When training a model with CTC loss, a maximum likelihood function is calculated and then optimized. This function is determined using the output of the network and the forward-backward algorithm, as described in the original paper [16]. While the process of calculating the maximum likelihood function is quite complicated, its use during training allows for a simple decoding process.

Part of CTC's simplicity can be attributed to its loss in generality. CTC can only be used for label sequence prediction tasks, not all sequence prediction tasks. This more-general set of all sequence prediction tasks includes those with continuous or ordinal output domains, such as audio or video generation, for which CTC is not applicable. Additionally, CTC inherently introduces the simplifying constraint that the ordering of elements in the input sequence and the corresponding labels in the output sequence must be the same. This is, of course, true for tasks such as speech recognition because words are always spoken in the order they are meant to be heard or written. However, this means that CTC is not suitable for tasks such as machine translation due to syntactic differences between languages (e.g. word order, subject-object-verb and object-adjective orderings, etc.). This constraint further reduces the set of tasks for which CTC may be used.

2.3.3 LSTM with CTC

The original paper on CTC specifies its use with recurrent neural networks (RNNs). This is a natural choice, since RNNs are inherently able to process variable-length sequences, and CTC takes a variable number of frames as input. Therefore, one can use an RNN model, such as a long short-term memory (LSTM) model, to output a variable-length sequence that can then be used by the CTC loss and decoding processes. Figure 2-14 shows the use of an LSTM model for temporal feature extraction and the use of the model's output by CTC for sequence prediction.

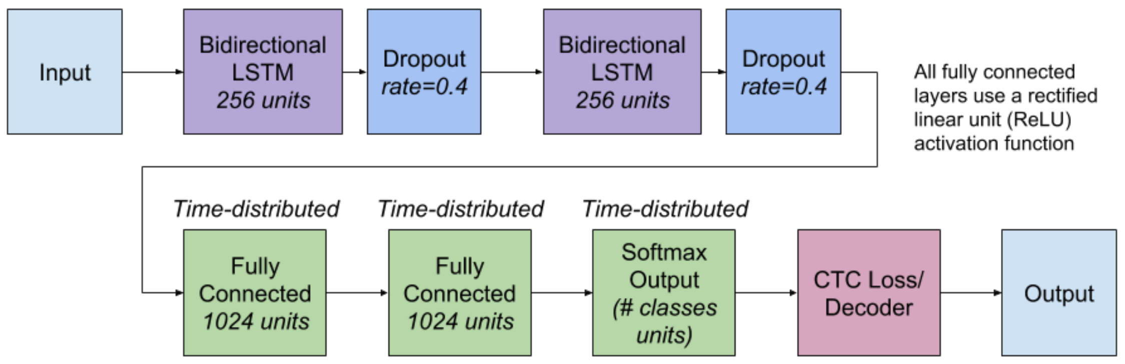
There is a major downside to using an LSTM, however, and that is its speed. Training an LSTM to a level capable of achieving similar results as other architectures takes drastically longer than those other architectures. Also, LSTMs, like all RNNs, tend to easily overfit to training data, which results in a model that fails to generalize well to unseen data. Despite LSTMs being the more natural choice for the task at hand, these downsides led us to also consider the use of CTC with other architectures. Nevertheless, the LSTM with CTC approach is compared to these other architectures in the experiments presented in this thesis. Reported error rates for the LSTM,



Sequence-to-sequence training/prediction
Matrix of class probabilities over time $[p_1, p_2, \ldots, p_n]$
during training
during prediction
CTC Loss
CTC Decoder
optimize
predict
Label sequence $[y_1, y_2, \ldots, y_k]$
Predicted sequence $[\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_k]$
Model output
Class probability vector p_1
Class probability vector p_2
Class probability vector p_n
Time-distributed fully connected layer
Fully Connected
Bidirectional LSTM layer
Initial state $f_{2,0}$
LSTM
$f_{2,1}$ $b_{2,n-1}$ $f_{2,2}$ $b_{2,n-2}$ $f_{2,n-1}$ $b_{2,1}$
Initial state $b_{2,0}$
Bidirectional LSTM layer
Initial state $f_{1,0}$
$f_{1,1}$ $b_{1,n-1}$ $f_{1,2}$ $b_{1,n-2}$ $f_{1,n-1}$ $b_{1,1}$
Initial state $b_{1,0}$
Model input
Sample x_1
Sample x_2
Sample x_n

Figure 2-14: The use of a bidirectional LSTM model for temporal feature extraction followed by sequence prediction using CTC. This figure shows a two-layer bidirectional LSTM with one fully connected layer, although it is common to use more layers.

however, are of an unoptimized LSTM model, due to the slow training time and lack of the amount of training data needed to properly train the model.

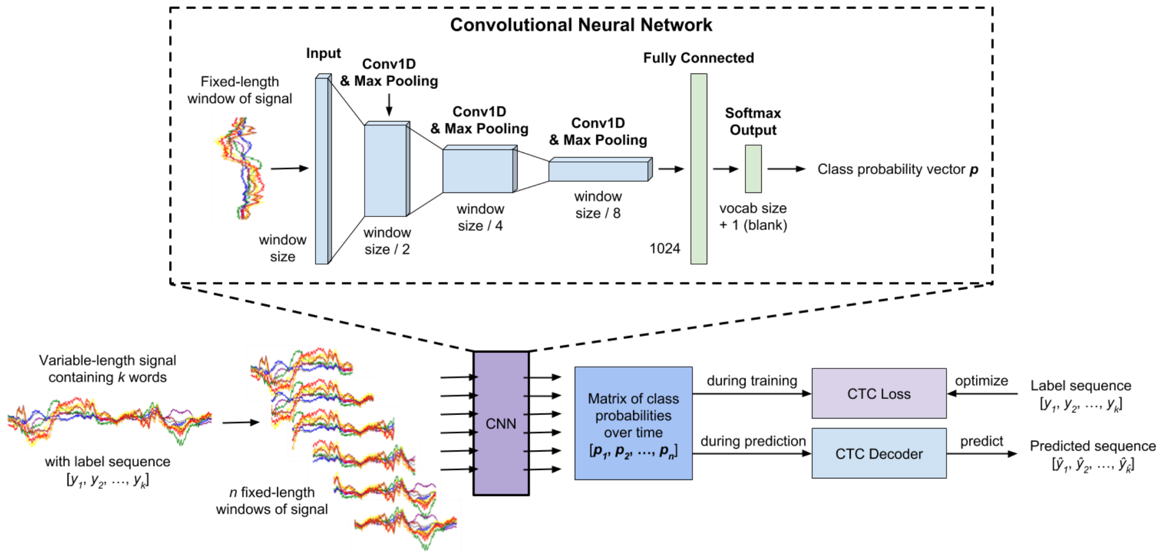


Input
Bidirectional LSTM 256 units
Dropout rate=0.4
Bidirectional LSTM 256 units
Dropout rate=0.4
All fully connected layers use a rectified linear unit (ReLU) activation function
Time-distributed
Fully Connected 1024 units
Time-distributed
Fully Connected 1024 units
Time-distributed
Softmax Output (# classes units)
CTC Loss/ Decoder
Output

Figure 2-15: The details of the LSTM architecture used in evaluation of an LSTM with connectionist temporal classification (CTC).

The LSTM architecture used in conjunction with CTC, as shown in Figure 2-15, uses two bidirectional LSTM layers of size 256 units, followed by dropout, two time-distributed fully-connected layers of size 1024 units, and a time-distributed softmax output layer. This multi-layer bidirectional LSTM encodes the variable-length input, producing a variable-length softmax output, which is then inputted to the CTC loss function and CTC decoder during training and prediction time, respectively.

2.3.4 CNN with CTC



Convolutional Neural Network
Fixed-length window of signal
Input
window size
Conv1D & Max Pooling
window size / 2
Conv1D & Max Pooling
window size / 4
Conv1D & Max Pooling
window size / 8
Fully Connected
1024
Softmax Output
vocab size + 1 (blank)
Class probability vector *p*
Variable-length signal containing *k* words
with label sequence $[y_1, y_2, \ldots, y_k]$
n fixed-length windows of signal
CNN
Matrix of class probabilities over time $[p_1, p_2, \ldots, p_n]$
during training
during prediction
CTC Loss
CTC Decoder
optimize
predict
Label sequence $[y_1, y_2, \ldots, y_k]$
Predicted sequence $[\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_k]$

Figure 2-16: The use of a CNN model for window feature extraction followed by sequence prediction using CTC. This figure shows a three-layer CNN with one fully connected layer and an output softmax layer, although it is common to use more layers.

Because the output of an RNN at a particular frame is directly dependent on the RNN's state as determined by previous frames, the output at each frame retains the context of all previous frames. The same is not true for convolutional neural networks (CNNs), which are generally used for spatial feature recognition and therefore have no internal state from past frames to use as context. However, we have found that with carefully selected hyperparameters, CNNs with CTC are able to achieve similar, if not better, results as LSTMs.

Despite our data being distinctly time series, the features can be learned as spatial features with a CNN, as shown in Figure 2-16. This does, however, require inputting a fixed-length window of the signal, as CNNs operate on fixed-size inputs only. Unfortunately, this need to split a variable-length signal into many fixed-length windows introduces two new hyperparameters, window size and window stride, and the tuning of both is critical in achieving good performance with a CNN with CTC model.

Because CNNs do not have memory to use as context like LSTMs, they must be capable of outputting an accurate probability distribution given only the input

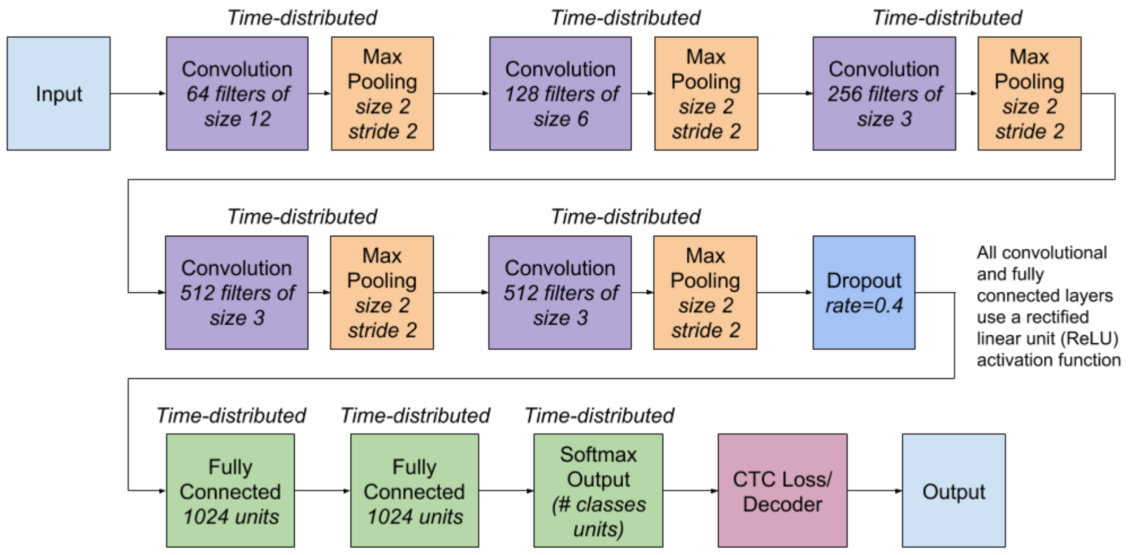

Time-distributed
Input
Convolution 64 filters of size 12
Max Pooling size 2 stride 2
Time-distributed
Convolution 128 filters of size 6
Max Pooling size 2 stride 2
Time-distributed
Convolution 256 filters of size 3
Max Pooling size 2 stride 2
Time-distributed
Convolution 512 filters of size 3
Max Pooling size 2 stride 2
Time-distributed
Convolution 512 filters of size 3
Max Pooling size 2 stride 2
Dropout rate=0.4
All convolutional and fully connected layers use a rectified linear unit (ReLU) activation function
Time-distributed
Fully Connected 1024 units
Time-distributed
Fully Connected 1024 units
Time-distributed
Softmax Output (# classes units)
CTC Loss/ Decoder
Output

Figure 2-17: The details of the CNN architecture used in evaluation of a CNN with connectionist temporal classification (CTC).

window. This requires that the window not be too small, lest the network be provided with too little information for classification. The window must also not be too large, lest it be too difficult for the network to determine which portion of a signal (that may span multiple words) is relevant for classification. The stride of the windows must also not be too large, or there may be too few frames corresponding to each label for CTC to accurately decode into an output sequence. Additionally, increases in either the window size or stride result in a weakly decreasing number of windows for CTC to work with, as shown in Equation 2.1. This is of concern because CTC requires that the input sequence be equal in length or larger than the output sequence.

$$n_{windows} = \left\lfloor \frac{l_{input} - l_{window}}{\text{stride}} \right\rfloor + 1 \tag{2.1}$$

The CNN architecture used in conjunction with CTC is a deep CNN consisting of five convolutional layers, each paired with a max pooling layer, followed by dropout, two fully-connected layers of size 1024 units, and a softmax output layer. All layers are time-distributed, as the input is a variable number of timesteps (i.e. signal windows). See Figure 2-17 for details on number of filters, filter sizes, strides, etc. These time-distributed convolutional layers encode the variable-length input, producing a variable-length softmax output, which is then inputted to the CTC loss function and CTC decoder during training and prediction time, respectively. The input window size for this model is one second of data (250 samples, given a 250 Hz sampling rate), and the window stride, determining the overlap of each timestep, is 250 milliseconds of data. These values were empirically chosen after careful tuning.

2.4 Language Model

Language models are probability distributions over a sequence of words that allow for the calculation of the probability of a given sequence of words. In turn, language models can be used to compare the probabilities of two or more sentences and determine the sentence with the highest likelihood. [18]. Sequence prediction models, specifically those related to speech recognition, sometimes make use of language models to increase the accuracy of predictions. Here I will discuss the use of a language model with connectionist temporal classification (CTC) to boost the accuracy of the CTC-based sequence prediction approaches.

CTC makes the assumption of independence of labels in the output sequence. This is a very strong assumption, and is clearly not true in the case of transcribing natural language. This assumption makes it impossible to directly integrate a language model into the CTC classification process. Fortunately, a language model can be applied to CTC externally in a very simple manner.

A paper by Chow et al. [19] discusses using a language model in conjunction with an "n-best" beam-search using the Viterbi algorithm. This is exactly the decoding process being performed by the CTC beam-search decoder, except that the input to the search is the output of the CTC classification model. Therefore, we can apply a language model simply by using the output of the CTC beam-search decoder. By outputting the top n most likely paths during decoding along with their likelihoods, we can then apply a language model to update the likelihoods and reorder the most likely paths. As long as a sufficient number of paths are outputted and the language model is representative of the silent speech being decoded, the language model is likely to lead to an increase in accuracy in the transcription process.

It is difficult to determine the probability of a sequence beyond a certain length directly from frequency counts, as the sequence may appear too infrequently for its frequency count to give an accurate estimate of its probability. Instead, we use the common approach of approximating this probability by analyzing the probabilities of subsequences of a certain length (i.e. n-grams). We approximate the probability of a length-n sentence using its length-k subsequences as shown in Equation 2.2.

$$p(w_1, w_2, ..., w_n) = \prod_{i=1}^{n} p(w_i | w_1, w_2, ..., w_{i-1})$$

$$p(w_i | w_1, w_2, ..., w_{i-1}) \approx p(w_i | w_{i-k}, w_{i-k+1}, ..., w_{i-1})$$

$$p(w_1, w_2, ..., w_n) \approx \prod_{i=1}^{n} p(w_i | w_{i-k}, w_{i-k+1}, ..., w_{i-1}) \tag{2.2}$$

We can use the frequency of length-k substrings in corpora of the target language to get a good approximation of the probabilities of each of the substrings and, in turn, the entire sentence, as shown in Equation 2.3.

$$p(w_i | w_{i-k}, w_{i-k+1}, ..., w_{i-1}) \approx \frac{\texttt{count}(w_{i-k}, w_{i-k+1}, ..., w_i)}{\texttt{count}(w_{i-k}, w_{i-k+1}, ..., w_{i-1})}$$

$$p(w_1, w_2, ..., w_n) \approx \prod_{i=1}^{n} \frac{\texttt{count}(w_{i-k}, w_{i-k+1}, ..., w_i)}{\texttt{count}(w_{i-k}, w_{i-k+1}, ..., w_{i-1})} \tag{2.3}$$

Language models can be used with other sequence prediction methods through use of the Viterbi algorithm to find the maximum likelihood output sequence from the model's output probability distributions. This thesis, however, focuses on applying language models using output from the CTC beam-search decoder, which already makes use of the Viterbi algorithm and therefore greatly simplifies integration of a language model.

Chapter 3

Datasets and Testing Procedures

In this chapter, I will discuss the datasets and testing procedures used in the evaluation of the continuous subvocal speech recognition system presented in this thesis.

3.1 Dataset Limitations

AlterEgo uses neuromuscular signals from internal articulation to transcribe silent speech into text. Before presenting the datasets used, I will discuss the downsides to using this specific modality as a means of input.

3.1.1 Lack of Applicable Datasets

There are many datasets of subvocal speech publicly available; however, none are specific to internal articulation – precisely the input modality that AlterEgo uses. One example is the EMG-UKA dataset [20], which consists of subvocalization data that includes mouth movements, as if speaking normally but without any sound. Internal articulation differs significantly from this means of input in that there are no visible movements. Any movement creates clearly visible features in electromyographic (EMG) data, and so the EMG-UKA dataset is not comparable to internal articulation data. The frequency range of features produced by large muscle movements, such as lip movement, overlaps significantly with the frequency range of internal articulation features, and therefore there is no effective way to separate the signals from these two actions. Due to the lack of applicable datasets, all datasets used in the evaluation of this system have been collected manually.

3.1.2 User and Session Dependence

Another challenge when working with data from internal articulation is that it is highly user-specific. The style of subvocalization may significantly differ across users, resulting in vastly different features for the same words. While the same is true for normal speech (e.g. accents and voice type affect the way one's speech sounds), there are several very large datasets that can be leveraged in order to learn this variation. The task of transcription from internal articulation, as previously mentioned, lacks the analogous datasets to learn from.

Additionally, data collected across several recording sessions, even by the same user, sometimes differs significantly. This may be for a few reasons. First, the data is very dependent on precise electrode placement. Even small perturbations in the positions of electrodes have been shown to lead to a decrease in accuracy. As electrodes are generally removed between recording sessions, it is expected that some amount of error is introduced between sessions due to electrode placement. Second, the data has also shown to be somewhat dependent on recording time. This is either due to one's style of subvocalization changing over time, or changes in one's physiological state (e.g. skin impedance, muscle fatigue, etc.) that cannot be controlled. For the reasons stated above, all experiments performed in the evaluation of this system were done so immediately after collection of the relevant training data. This, unfortunately, means that our models must be trained with a relatively small number of training samples, as will be discussed in the next section. We expect that given the opportunity to collect larger datasets, the accuracy of the system could be improved.

3.2 Recorded Datasets

In the evaluation of this system, we use the standard digit classification benchmark task for subvocal transcription systems, as well as classification tasks focused on vocabularies of selected words. For both vocabularies, we test our classification model with a single-label classification task (i.e. labelling individual words), as well as our system as a whole with our primary task, label sequence (entire sentence) prediction. Additionally, we explore the potential of phoneme- and syllable-level prediction by mapping the recorded words to smaller units of speech, and address the challenges facing a system operating on these levels.

3.2.1 Digit Sequence Datasets

A standard benchmark for subvocal speech transcription systems is testing the classification accuracy on subvocal digits 0 through 9. We recorded multiples datasets of subvocal digits, each containing 50 samples per digit. This serves as a baseline to determine the word-level classification accuracy for subvocal digits, a useful baseline when evaluating a sentence-level system. Then, we collected two datasets of subvocal digit sequences – the first containing all permutations of digits 0, 1, and 2 as a simple sequence-level test; and the second containing random sequences of five digits (chosen from digits 0 through 9).

3.2.2 10-Word Vocabulary, 20-Sentence Dataset

We also recorded datasets of subvocal words from a small vocabulary of 10 selected words. The selected words focused on communicating basic needs: [`"i"`, `"am"`, `"cold"`, `"hot"`, `"hungry"`, `"tired"`, `"want"`, `"need"`, `"food"`, `"water"`]. Similarly to the subvocal digits datasets, we first test the world-level classification accuracy as a baseline. We then construct 20 sentences, each containing five words chosen from the 10-word vocabulary, as shown in Table 3.1. The dataset consists of 10 samples of each of these 20 sentences (we are able to use so few samples per sentence because the sentences share many of the same words). No significance is placed on constructing grammatical

Selected 10-Word Vocabulary				
i	am	cold	hot	hungry
tired	want	need	food	water

Recorded (Nonsensical) Sentences	
cold water want hungry i	want hot cold hungry am
water tired hot i food	hungry want am water cold
food hungry need hot tired	i need cold tired food
cold hot want water am	want food i hot water
need want tired cold am	food cold need am i
hot food tired am hungry	tired hungry food am hot
hungry water need i want	need hungry cold want i
water hungry tired need food	cold food want need water
food hot am cold i	am tired water hot hungry
water i am need tired	tired i hungry hot need

Table 3.1: The 10 vocabulary words and the 20 sentences constructed from the 10 word vocabulary such that no two sentences share a bigram. The sentences are nonsensical, but this is acceptable (even somewhat desirable) for the task of learning the alignment between individual words. Example recordings of these sentences can be seen in Figure B-1 in Appendix B.

English sentences, as the vocabulary can easily be replaced with other words. The goal of testing this dataset is simply to see if sentences that have never been seen before can be reliably transcribed. Example recordings of these sentences can be seen in Figure B-1 in Appendix B.

When constructing the 20 sentences, we add the constraint that no two sentences can share a bigram (i.e. two-word phrase), which, of course, means that no two sentences can share any higher level n-grams as well. This is done intentionally to ensure that the CTC classifier does not learn arbitrary alignments between groups of words, instead of the correct alignment of the individual words. For example, Figure 3-1 shows an example training sentence ("i need cold tired food"). If in all training sentences every occurrence of the word "i" is followed by "need" and every occurrence of the word "need" is preceded by "i", then incorrect alignments such as those shown in the figure may be learned. This is because there is no training data that forces the correct alignment between this group of words ("i need") to be learned. Training on other sentences that include the same words but none of the same bigrams (e.g.

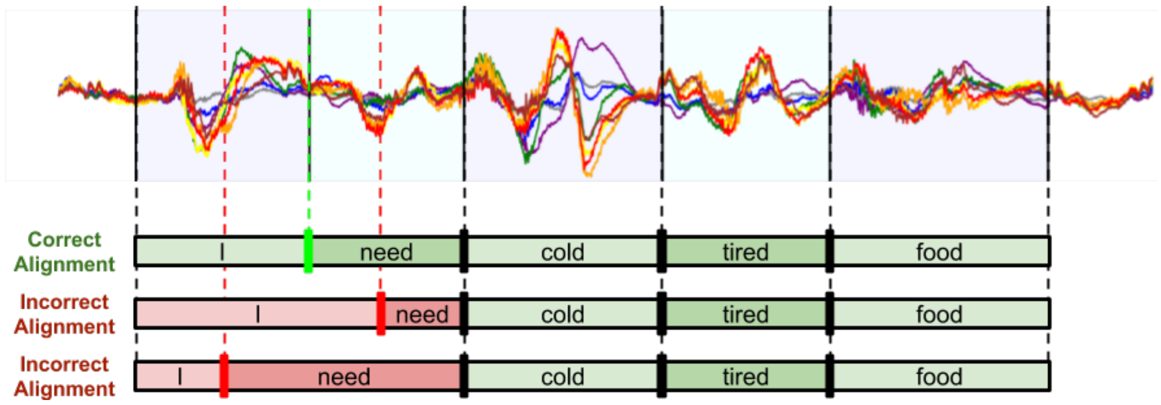


Figure 3-1: In this example, if in all training sentences every occurrence of the word "i" is followed by "need" and every occurrence of the word "need" is preceded by "i", then incorrect alignments such as those shown above may be learned.

"need i", "you need", "i want", etc.) is one way to ensure the correct alignment is learned.

3.2.3 20-Word Vocabulary, 200-Sentence Dataset

We also wanted to test our model on a larger vocabulary, sentences with a variable number of words, and a higher rate of speech than the previous datasets. For this, we created a 20-word vocabulary, focused on communicating basic needs and asking simple questions: ['hello', 'i', 'am', 'you', 'are', 'the', 'want', 'need', 'cold', 'hot', 'food', 'where', 'what', 'how', 'feeling', 'doing', 'tired', 'water', 'hungry', 'thirsty']. From this vocabulary, we generated 200 sentences, each consisting of 3-6 words. Samples from 150 of the sentences were used in training, and samples from the remaining 50 sentences were used exclusively in a test set of sentences that were never seen during training. Similarly to the 10-word vocab dataset, when generating these sentences, we prioritized sentences that share very few bigrams with other sentences (although there is no hard constraint, as was the case with the 10-word vocab dataset). This means that not only are the sentences in the test set entirely absent from the training set, but the two sets share very few sub-sentence phrases as well. We collected 5 samples of each sentence, split between the training and test sets, summing to 1000 samples collected in total. Table 3.2 shows the 20 chosen vocabulary words. The list

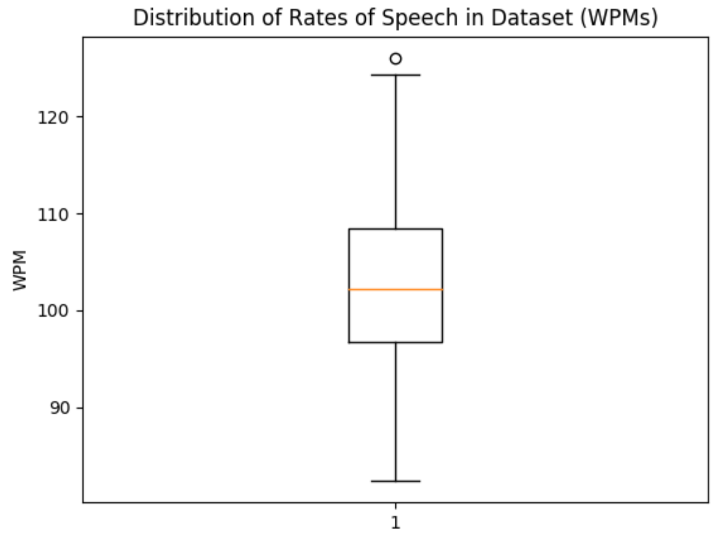


Figure 3-2: The distribution of rates of speech in the 20-word vocabulary, 200-sentence dataset, reported in words per minute (WPM). 1000 samples of sentences were collected in total. The average is 102.4 WPM, with 62% of samples having a rate of speech of at least 100 WPM.

of the 200 sentences we generated can be found in Table A.1 in Appendix A. Also, example recordings of some of these sentences can be seen in Figure B-2 in Appendix B.

Selected 20-Word Vocabulary									
hello	i	am	you	are	the	want	need	cold	hot
food	where	what	how	feeling	doing	tired	water	hungry	thirsty

Table 3.2: The 20 vocabulary words chosen to make up the 200 sentences. The 200 sentences we generated from this vocabulary can be found in Appendix A.

While recording this dataset, the user was given feedback regarding their rate of speech, reported in words per minute (WPM). Using this feedback, they were instructed to try to achieve a consistent rate of speech of 100 WPM, or slightly higher. The average rate of speech for the entire dataset is 102.4 WPM, with 62% of samples having a rate of speech of at least 100 WPM. The distribution shown in Figure 3-2. This is in sharp contrast to the other sentence-level datasets we collected (without any attention to rate of speech), which averaged 70.5 WPM.

3.2.4 Subvocal Phonemes and Syllables

We also explored the potential of subword-level sequence prediction, such as phoneme- and syllable-level prediction. Because a CTC-based sequence prediction model does not require segmented input data, we used the same dataset as for the word-level sequence prediction tasks and simply mapped the sequences of words to the sequences of their constituent phonemes/syllables. Unfortunately, we found that it was quite difficult to train a model to predict these smaller units of speech, as few distinguishable features appear in the data. Those features that do appear have a much longer duration than a single phoneme or syllable, and so are representative of a larger unit of speech and not suitable for training at the level of phonemes or syllables.

3.3 Testing Procedures

We used various testing procedures in evaluating the quality of the recorded datasets and the accuracy of our system. Due to the session dependence affecting our data, testing data was collected during the same recording session as the training data (generally immediately afterwards). For this reason, in addition to testing the system in real-time after training our models, we also recorded and saved the real-time testing data to file, which provided us with a test set that we can evaluate directly during training as well as playback later on (as if in real-time) after changes to the system have been made. We saved the entire recording session, rather than only the sequences representing subvocal speech, so that we can adjust the bounds of the labeled data later if necessary. Simulating real-time tests allowed us to make changes to the model and test it as if in real-time without needing to collect new training data each time. It has also allowed us to test our models on subsets of the channels used during data collection. This is useful for determining which electrodes provide relevant information to our models. For the sake of consistency, most of the experiments reported in Chapter 4 use test sets evaluated during training time.

Chapter 4

Evaluation

In this chapter we report the error rates achieved by each of the approaches (as outlined in Chapter 2) for each of the collected datasets. We also evaluate the error rate reduction that results from the use of a language model, calculate the rate of information transfer, where appropriate, and report the results of an experiment to determine the relative contributions of each electrode to the classification accuracy of our model.

4.1 Metrics

First, we will discuss the metrics used in evaluating the accuracy and information transfer rate of our sequence prediction models.

4.1.1 Average Normalized Edit Distance

In evaluating the accuracy of sequence prediction models, we need a measure of how close to correct a predicted sequence is to the target sequence. Because the predicted sequence can vary in length, we cannot use a simple elementwise comparison to calculate accuracy. Instead, the normalized edit distance between the predicted sequence and target sequence is used as the metric for sequence prediction error rate. In other words, the error rate is defined as the number of edits (replacements, additions, or deletions) to the predicted sequence needed to arrive at the target sequence, divided by the length of the target sequence. The average of this over an entire dataset is used to evaluate our models. Equation 4.1 gives a more formal definition of the average normalized edit distance as a function of the predicted sequences $[\hat{\mathbf{y_1}}, \hat{\mathbf{y_2}}, ..., \hat{\mathbf{y_n}}]$ and the corresponding target sequences $[\mathbf{y_1}, \mathbf{y_2}, ..., \mathbf{y_n}]$.

$$\delta = \frac{1}{n} \sum_{i=0}^{n} \frac{\text{levenshtein}(\hat{\mathbf{y}}_\mathbf{i}, \mathbf{y_i})}{|\mathbf{y_i}|} \tag{4.1}$$

We use a length-normalized metric so that the length of the target sequences used in evaluation does not affect the report error rate. This allows us to evaluate our model with variable-length sequences, as we do in our final dataset (see Section 3.2.3).

4.1.2 Rate of Speech and Information Transfer Rate

The primary goal of this system is to transcribe internal articulation continuously at high rates of speech. Therefore, while we evaluate the error rates for each of our models as a measure of accuracy, it is also important to evaluate the rate of speech and information transfer.

First, we calculate the rate of speech of a dataset by calculating the average words per minute (WPM) of the sentences. This is done as in Equation 4.2, where s is the sampling rate in Hz, l_i is the number of samples in sentence i, and k_i is the number of words in sentence i for each of the n sentences.

$$\text{WPM} = \frac{60s}{n} \sum_{i=0}^{n} \frac{k_i}{l_i} \tag{4.2}$$

This metric gives the rate of speech of the data, but tells us nothing about the system's performance at this rate of speech. The performance of human-computer interfaces (specifically, brain-computer interfaces) is generally measured by bit-rate (also referred to as Wolpaw rate), as explained by Kronegg et al. [21] and originally proposed by Wolpaw et al. [22]. The bit-rate of a system is calculated as shown in Equation 4.3, where B is the information transfer rate in bits per minute, V is the application speed (average words per minute, in our case), P is the classification accuracy (one minus the word error rate, in our case), and N is the vocabulary size.

$$B = V \left[\log_2 N + P \log_2 P + (1 - P) \log_2 \left(\frac{1 - P}{N - 1} \right) \right] \tag{4.3}$$

This metric takes into account not only the rate of speech, but also the accuracy and size of the vocabulary. Larger vocabularies allow for the transfer of more information with a single word, since there are more unique symbols to choose from. At the same time, a less accurate system is less capable of transferring information due to errors. Therefore, maximizing this metric, for any system, requires finding a balance between vocabulary size and accuracy, which are generally inversely correlated.

4.2 Model Comparison

4.2.1 Digit Sequence Datasets

The system was first tested with a very simple task – sequence label prediction for permutations of the digit sequence [0, 1, 2]. With a vocabulary of only three words, this test was primarily used for evaluating the effectiveness of CTC in sequence prediction given near-perfect word classification accuracy, which is to be expected for such a small vocabulary. We confirmed this expectation by evaluating our network on the corresponding single-label classification task, resulting in a 97.3% test accuracy.

The dual classifier approach, as described in Section 2.3.1, achieves an error rate (i.e. average normalized edit distance) of 11.8%. Most of this error is attributed to misclassifications, although false positives and false negatives on the part of the activation classifier were the cause of 1.7% and 2.5% of the errors, respectively.

The CNN with connectionist temporal classification (CTC), as described in Section 2.3.4, achieves a lower error rate of 5.5%. Interestingly, the largest source of error in this case is not misclassification, but rather erroneous additions or deletions to the label sequence. This can be seen in Table 4.1, which shows that out of five incorrect predictions, three of the errors are erroneous additions or deletions, and the other two are caused by misclassifications.

Target	Prediction
0, 2, 1	0, 2, 1
2, 1, 0	2, 1, 0
2, 0, 1	2, 0, 1
0, 2, 1	0, 2, 1
2, 0, 1	2, 0, 1
2, 1, 0	**0**, 2, 1, 0
1, 0, 2	1, 0, **1**
0, 1, 2	0, 1, 2
2, 1, 0	2, 1, 0
2, 1, 0	2, 1, **2**

Target	Prediction
2, 1, 0	2, 1, 0
0, 1, 2	0, 1, 2
0, 1, 2	0, 1, 2
1, 0, 2	1, 0, 2
1, 2, 0	1, 2, 0
0, 2, 1	0, 2, 1
2, 0, 1	2, 0, 1
1, 2, 0	1, 2, 0
1, 0, 2	1, 0, 2
0, 1, 2	0, 1, 2

Target	Prediction
0, 2, 1	0, 2, 1
0, 1, 2	0, 1, 2
0, 2, 1	0, 2, 1
1, 0, 2	1, **2**, 0, 2
1, 0, 2	0, 2
1, 2, 0	1, 2, 0
1, 2, 0	1, 2, 0
2, 0, 1	2, 0, 1
2, 0, 1	2, 0, 1
1, 2, 0	1, 2, 0

Table 4.1: Example test output of the 0-1-2 permutation sequence labeling task. Incorrect predictions and the corresponding target sequences are bolded.

The LSTM with CTC, as described in Section 2.3.3, unfortunately takes drastically longer than the CNN to train – on the order of days – and for this reason could not be optimized for the dataset. The unoptimized LSTM with CTC has been shown to achieve an error rate of 15.5%. For larger datasets, this problem is exacerbated. Therefore we only report the results of the CNN for the other datasets.

The system was then tested on the more difficult task of labeling sequences of digits consisting of digits 0 through 9. All sequences tested were 5 digits in length, meaning this task evaluates the model's ability to label longer sequences comprised of a larger, 10-word, vocabulary than the previous task. We again evaluate our models on the corresponding single-label classification task, which results in a 96.1% test accuracy, as shown by the confusion matrices in Figure 4-1.

The dual classifier performed similarly on this larger-vocabulary dataset, achieving an average normalized edit distance of 12.6%. The CNN with CTC approach achieved a minimum error rate of 8.5%. This slightly higher error rate (compared to the smaller digits dataset), while possibly attributable to variation in dataset quality, is likely the result of CTC having difficulty learning the alignment between words given a larger vocabulary. Because the model must learn to determine word alignment at the same time that it learns to classify individual words, a larger vocabulary likely results in a

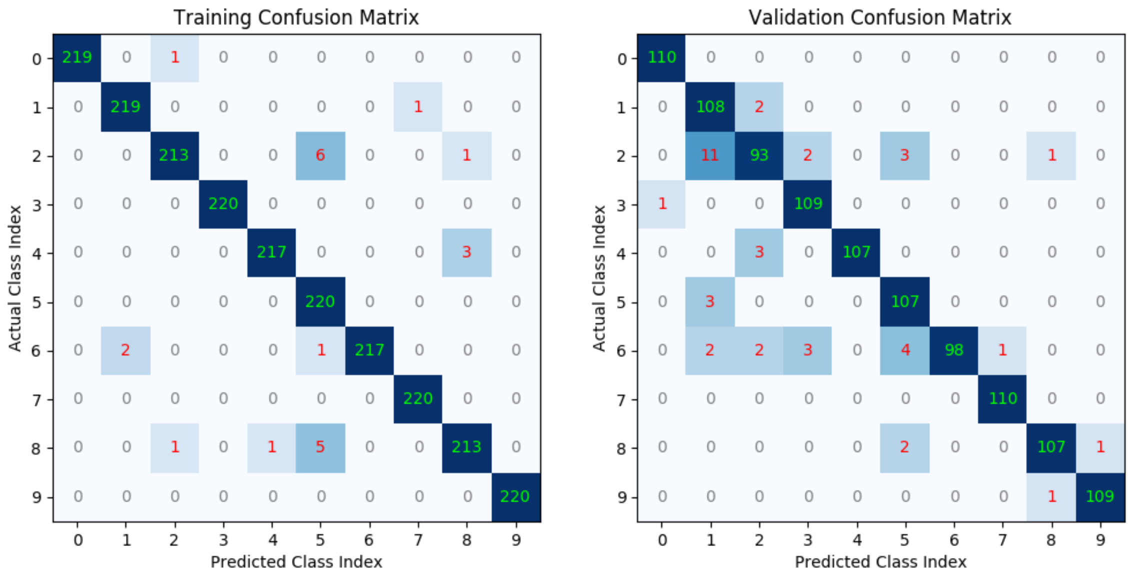


Figure 4-1: The training and validation confusion matrices for the task of classifying digits 0 through 9.

higher error rate even if a single-label classifier performs similarly on the larger vocab.

4.2.2 10-Word Vocabulary, 20-Sentence Dataset

Next, we evaluated our system with sentences from a vocabulary of 10 selected words. The selected words focused on communicating basic needs: ["i", "am", "cold", "hot", "hungry", "tired", "want", "need", "food", "water"]. The single-label classification of this vocabulary resulted in a 93.3% test accuracy. This is noticeably less than the 10-digit vocabulary evaluated in the previous section, and may be indicative of the words in this vocabulary being less distinct from each other. The higher error rates of the sequence-to-sequence models seem to suggest the same. The dual classifier achieves an error rate of 17.3%, and the CNN with CTC was able to achieve an error rate of 11.5% without the use of a language model (as was the case in the previously reported experiments). However, it was able to achieve a much lower error rate of 7.0% with the use of a language model (see Section 4.3 for more details on the language model evaluation). Table 4.2 shows example test output from the CNN with CTC model.

Target	**Prediction**
cold food want need water	cold food want need water
water hungry tired need food	water hungry tired need food
tired i hungry hot **need**	tired i hungry hot
water hungry tired need food	water hungry tired need food
water hungry tired need food	**want** hungry tired need food
i need cold tired food	i need cold tired food
water hungry tired need food	water hungry tired need food
cold food want need water	cold food want need water
tired i hungry hot need	tired i hungry hot need
i need cold tired food	i need cold tired food
food **hungry need** hot tired	food **tired food** hot tired
water hungry tired need food	water hungry tired need food

Table 4.2: Example test output of the 10-word vocabulary sequence labeling task. Incorrect predictions and the corresponding target sequences are colored red.

4.2.3 20-Word Vocabulary, 200-Sentence Dataset

We also evaluated our system with variable-length sentences from a vocabulary of 20 selected words. For this dataset, we aimed for a rate of speech of ~100 WPM, achieving an actual average rate of speech of 102.4 WPM – significantly higher than the average 70 WPM of the other datasets. The selected words focused on communicating basic needs and asking simple questions: ['hello', 'i', 'am', 'you', 'are', 'the', 'want', 'need', 'cold', 'hot', 'food', 'where', 'what', 'how', 'feeling', 'doing', 'tired', 'water', 'hungry', 'thirsty'].

The single-label classification of this vocabulary resulted in a 82.7% test accuracy. The dual classifier achieves an error rate of 26.2%. Even though this vocabulary is larger than that of other datasets, and so a decrease in accuracy is expected, the observed accuracy of the dual classifier approach is well below what is required for a practical system. The CNN with CTC approach, on the other hand, was able to achieve an error rate of 14.8% without the use of a language model. Similar to the 10-word vocab dataset, the model performs even better with the use of a language model, achieving an error rate of 10.7% (see Section 4.3 for more details on the language model evaluation). A word error rate of 10.7% makes for a much more usable system, especially at a rate of above 100 WPM. Furthermore, calculating the

Target	**Prediction**
how cold water	how cold water
hot hello are thirsty	hot hello are thirsty
what are the i feeling tired	what are the i feeling tired
thirsty feeling you hot water **hello**	thirsty feeling you hot water
cold feeling hungry	cold feeling hungry
how hot food	how hot food
where **hot** food	where **i** food
need you what	need you what
feeling hungry **where** food thirsty	feeling hungry **am** food thirsty
you want what	you want what
what feeling how want water where	what feeling how want water where
feeling what am tired	feeling what am tired

Table 4.3: Example test output of the 20-word vocabulary sequence labeling task. Incorrect predictions and the corresponding target sequences are colored red.

information transfer rate using a 10.7% WER and a rate of speech of 102.4 WPM results in a bit-rate of 345.8 bits per minute – significantly higher than other similar systems, and drastically higher than BCI systems, which generally report between 5-25 bits per minute according to Kronegg et al. [21]. Table 4.3 shows example test output from the CNN with CTC model.

4.3 Language Model

For certain vocabularies, the use of a language model can significantly reduce the error rate of a sequence-to-sequence model. To test the use of a language model with our system, we used a simple one based on the bigrams and trigrams of the sentences in the dataset of interest. To apply the language model, we halved the probability of a sequence for each bigram and trigram it contained that has a frequency count of zero in our language model. In general, language models leverage assumptions or knowledge known about the dataset to increase accuracy. In our case, we make the simple assumption that bigrams and trigrams that are unseen in the dataset are half as likely to occur as those that are found in the dataset, and evaluate the CNN with CTC model accordingly.

Figure 4-2 shows the reduction in error rate achieved upon applying the language

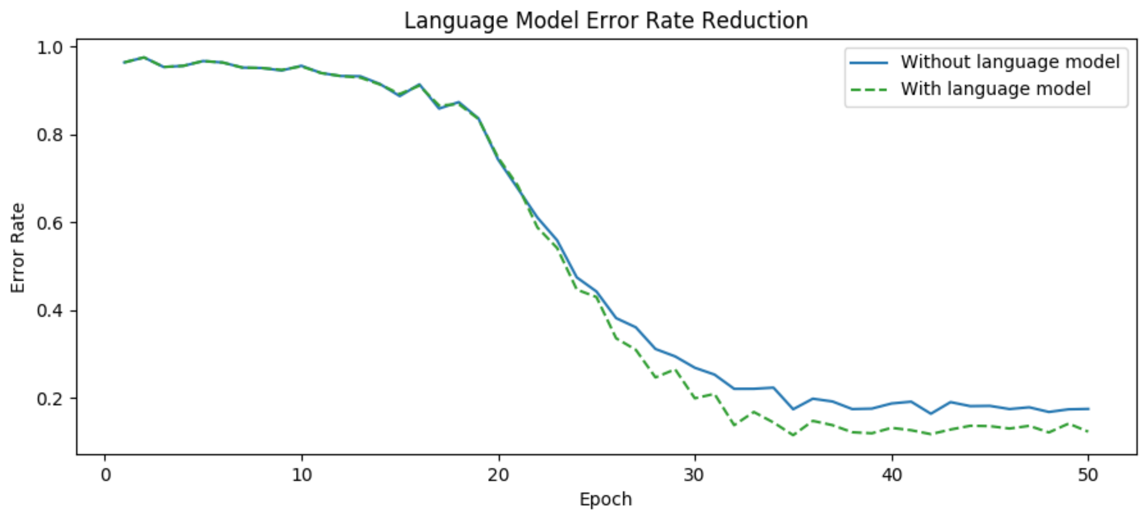


Figure 4-2: Comparison of error rates of the CNN with CTC model with and without a language model.

model described above, and how the language model affects the error rate throughout the training process. As expected, the language model introduces little benefit until the model is moderately-well trained (around epoch 25). This is because a language model is unlikely to lead to much of an improvement if the classifier is not already somewhat close to the correct prediction. After the model is moderately-well trained, however, the language model reduces the error rate by a consistent 4-15%. For the 10-word vocab dataset, this results in a final error rate of 7.0% – a significant reduction from the 11.5% error rate without the use of a language model. Similarly, for the 20-word vocab dataset, this results in a final error rate of 11.4%, rather than the 16.4% error rate without the use of a language model

In using CTC, we always choose the most-probable prediction, according to the probabilities reported by the CTC decoder, as the final prediction. Therefore, looking at the error rate as determined by only the most probable prediction, we fail to see some of the benefit of the language model – namely, the sequences that were closer to being predicted correctly, but still weren't. Another useful method of evaluating the boost in accuracy provided by a language model is to look at the error distribution over the top-k highest probability predictions. For this, we can look at the top-k error rate, defined as the average minimum error rate among the k most probable

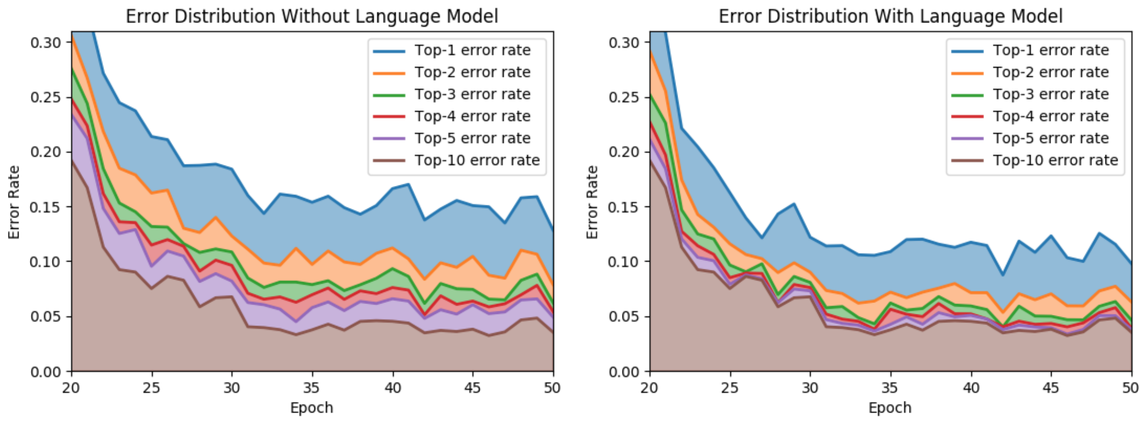


Figure 4-3: Comparison of error distributions of the CNN with CTC model with and without a language model when near fully trained. Note that the top-10 error rates with and without a language model are the same. The differences between these top-k error rates give a measure of where in the k most probable predictions the highest-accuracy predictions reside. For example, a large difference between the top-k and top-(k+1) error rates indicates that some (k+1)th most probable predictions are higher in accuracy than the corresponding kth most probable predictions. This, in turn, signals that the accuracy of the system could be improved if the probabilities of the predictions could be better correlated with their accuracies.

predictions, for several values of k. The differences between these top-k error rates give a measure of where in the k most probable predictions the highest-accuracy predictions reside. For example, a large difference between the top-k and top-(k+1) error rates indicates that some (k+1)th most probable predictions are higher in accuracy than the corresponding kth most probable predictions. This, in turn, signals that the accuracy of the system could be improved if the probabilities of the predictions could be better correlated with their accuracies. As one can see in Figure 4-3, the differences between the top-k error rates for higher values of k are much smaller when a language model is used, indicating that the more-accurate predictions are ranked closer to the top in the ordering of most-likely predictions when using a language model. This confirms that the language model is effective in providing a boost in accuracy, even for the predictions that are not most-probable and therefore are not used.

Table 4.4 shows the top 10 predictions for an example sequence with and without a language model. The predictions are colored green, yellow, or red, depending on

Color Legend: 🟩 edit dist. of 0 (correct) 🟨 edit dist. of 1 🟥 edit dist. of 2

Ground truth: food hungry need hot tired

	Top 10 predictions without LM	**Top 10 predictions with LM**
#1	🟨 food hungry need want tired	🟩 food hungry need hot tired
#2	🟩 food hungry need hot tired	🟨 food hungry need want tired
#3	🟨 food hungry need want tired	🟩 food hungry need hot tired
#4	🟥 food hungry need want water tired	🟨 food hungry need want tired
#5	🟥 food hungry water need want tired	🟨 food hungry need water tired
#6	🟥 food hungry tired need want tired	🟥 food hungry water need want tired
#7	🟥 food hungry need want i tired	🟥 food hungry tired need want tired
#8	🟩 food hungry need hot tired	🟥 food hungry need want water tired
#9	🟥 food hungry i need want tired	🟥 food hungry i need want tired
#10	🟨 food hungry need water tired	🟥 food hungry need want i tired

Table 4.4: The top 10 predictions with and without a language model, in order of decreasing probability, for an example sequence. The predictions are colored green, yellow, or red, depending on if they have an edit distance from the target sequence of 0, 1, or 2, respectively. This example is an instance of the most probable prediction being correct only with the use of a language model. Note: there are many repetitions, since the CTC decoder outputs each prediction independent of all others.

if they have an edit distance from the target sequence of 0, 1, or 2, respectively. One can clearly see that the application of a language model often results in correct predictions being more probable, and therefore more highly ranked within the top 10 predictions. In fact, this example is an instance of the most probable prediction being correct only with the use of a language model. This is just one example, however, and there are many sequences for which the predictions are already correct without a language model, as well as some for which the language model doesn't lead to any significant improvement.

4.4 Electrode Contribution

The electrode positions used for experiments in this thesis were chosen empirically. Experimenting with various subsets of the 8 chosen positions, which are shown in the left plot of Figure 4-4, has revealed that not all of the electrodes are necessary. In fact, many subsets of the electrodes achieve similar accuracies, likely due to the fact that

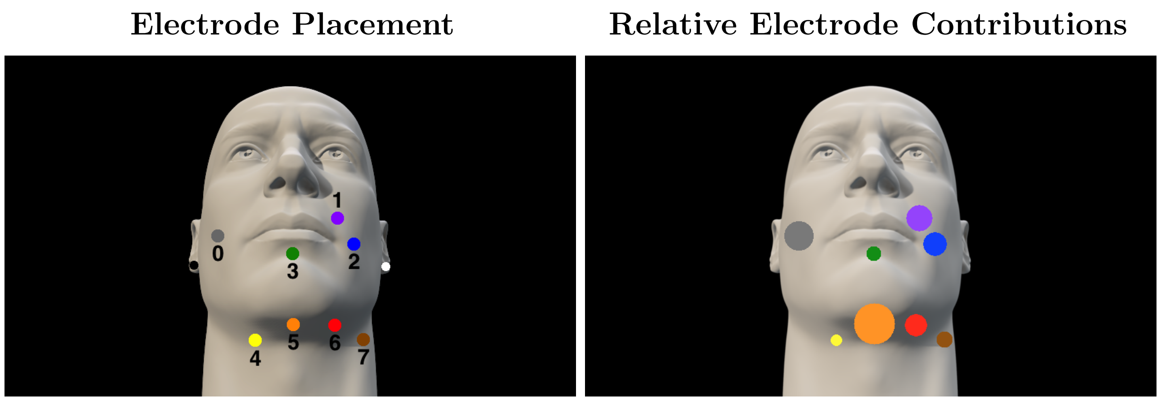


Figure 4-4: Left: The arrangement of 8 signal electrodes (colors and gray, on face and neck), with reference and bias electrodes also shown (white and black, on earlobes). Right: The relative electrode contributions, represented by the sizes of the electrode markers.

many of the electrodes pick up very similar features. For this reason, we wanted to quantify how large of a role each electrode played in determining the overall accuracy of our models. To do this, we calculated the contribution of each electrode to the classification accuracy for a specific task (digit prediction) by running over 1000 trials with different subsets of electrodes. The contribution of an electrode x is defined as the average percent decrease in error rate upon adding the electrode x to all subsets of electrodes that do not already include x. More formally, the contribution of each electrode is calculated as shown in Equation 4.4.

$$\text{the set of all electrodes,} \quad A := \{0,1,2,3,4,5,6,7\}$$

$$\delta(C) := \text{the average error rate achieved with electrode set } C \subset A$$

$$\text{contribution of } x \in A, \quad \alpha(x) := \frac{1}{2^{|A|-1}} \sum_{C \subset (A - \{x\})} \frac{\delta(C) - \delta(C \cup \{x\})}{\delta(C)} \tag{4.4}$$

Using this method, we arrived at the results shown in Table 4.5, which are also shown visually in the right plot of Figure 4-4. Unsurprisingly, electrodes 5, 1, and 2, all of which show high amplitude features, have high contribution. Electrode 5 contributes the most by far, on average reducing the error rate to less than half

Electrode 0	**Electrode 1**	**Electrode 2**	**Electrode 3**
0.2858	0.2243	0.1818	0.0715

Electrode 4	**Electrode 5**	**Electrode 6**	**Electrode 7**
0.0434	0.5543	0.1602	0.0879

Table 4.5: The calculated electrode contributions, shown visually in Figure 4-4.

of its original value (55.4% decrease) upon inclusion. Surprisingly, electrode 0 also contributes significantly (second highest) to the accuracy, which may be because it is located far away from the other electrodes, and therefore may pick up a significantly different signal. Also surprisingly, electrode 3 contributes very little, despite picking up a signal that seems fairly distinct from that of the other electrodes. This may indicate that the features collected by electrode 3, while distinct, simply don't help the model distinguish between words as much as the features from other electrodes.

Chapter 5

Conclusion

5.1 Summary of Results

In this thesis, we test several approaches to label sequence prediction against multiple self-collected datasets. The first approach uses a dual word and activation classifier to classify words and the silence between them in order to output a sequence of individual labels. Despite performing fairly well, this is arguably a naive approach, and one that requires pauses between words, resulting in a slow rate of transcription. The second approach explored uses an LSTM with connectionist temporal classification (CTC), which is a method of unsegmented label sequence prediction that uses an augmented single-label classifier and a beam-search decoder. However, we were unable to fully evaluate this approach due to the slow training time characteristic of LSTMs and the difficulty of training LSTMs without very large datasets. Our final and highest-performing approach was a carefully-tuned CNN with CTC. We also include a language model that uses the output of CTC's beam-search decoder to further reduce error rates.

Table 5.1 compares the performance of each approach when evaluated on each of the tasks. We report the single-label classification accuracy of a simple CNN on each of the datasets (for use as a single-label baseline), as well as the sequence prediction error rates (i.e. average normalized edit distance) for all of the sequence-to-sequence approaches. As expected, higher accuracies on the single-label classification tasks

	Single-Label Accuracy	Sequence Prediction Error Rate (average normalized edit distance)			
Dataset	**CNN**	**Dual Class.**	**LSTM+CTC**	**CNN+CTC**	**CNN+CTC+LM**
{0,1,2} sequences	0.973	0.118	0.155	0.055	N/A
10-digit sequences	0.961	0.126	-	0.085	N/A
10-word, 20-sent.	0.933	0.173	-	0.115	0.070
20-word, 200-sent.	0.827	0.262	-	0.148	0.107

Table 5.1: The final accuracy/error rates achieved by each approach, evaluated against each of the collected datasets. We report the single-label classification accuracy of a simple CNN on each of the datasets (as a single-label baseline), as well as the sequence prediction error rates (i.e. average normalized edit distances) for all of the sequence-to-sequence approaches. While the LSTM with CTC approach was tested on all three of the datasets presented here, drastically longer training times and the lack of large datasets prevented us from fully training and optimizing the LSTM. Therefore, we only report the error rate of the unoptimized LSTM approach for one dataset.

were observed along with lower error rates on the corresponding sequence prediction tasks. For all of the datasets evaluated, the CNN with CTC model outperformed the dual classifier approach significantly – achieving a 6.9% lower error rate on average. While the LSTM with CTC approach was also implemented and ran on all of the datasets presented here, drastically longer training times and the lack of large datasets prevented us from fully training and optimizing the LSTM as we did with the other models. Therefore, while we still report the error rate of the unoptimized LSTM approach for one dataset, we will not draw any conclusions regarding the error rate of this approach.

Furthermore, a language model can be applied to the 10-word and 20-word vocab sentence datasets, unlike the digits datasets. The integration of a simple language model to the CNC with CTC approach further lowered the error rate of the 10-word vocab dataset by 4.5%, achieving a final error rate of 7.0%. Similarly, the language model reduced the error rate of the 20-word vocab dataset by 4.1%, resulting in a final error rate of 10.7%. This amounts to a more than 10% and 15% error rate reduction between the dual classifier and the CNN with CTC and language model for the two sentence datasets, respectively.

The use of CTC to determine word alignment, rather than the activation classifier used by the dual classifier approach, removes the need to have a significant pause between words. As such, the potential transcription rate of models using CTC is significantly higher. The average rate of speech for the first three collected datasets is about 70 words per minute (WPM), and the rate of speech for the 20-word, 200-sentence dataset is a much higher 102.4 WPM. CTC is capable of learning from such high rates of speech without noticeably sacrificing accuracy. In fact, many of the correctly-transcribed sentences in the last dataset have a rate of speech of above 110 or even 120 WPM, demonstrating that the CTC-based approach is capable of learning from rates of speech significantly higher than 100 WPM. The maximum transcription rate we found the dual classifier is capable of, on the other hand, is only about 40 words per minute. Calculating the information transfer rate for the CNC with CTC model on the final dataset, which gives a 10.7% error rate at an average rate of speech of 102.4 WPM, results in a bit-rate of 345.8 bits per minute.

5.2 Conclusion and Discussion

In this thesis, I present my work on a continuous silent speech recognition system, capable of transcribing neuromuscular signals from internally articulated sentences. The system is able to accurately transcribe silent speech at rates of over 100 words per minute (WPM). As this is only slightly lower than the rate of conversational speech, the system does not require significant pauses between words, and feels fairly natural to use as a result. The largest dataset that we collected for the evaluation of our system consists of 200 unique sentences (5 recorded samples each) comprised of 3-6 words from a 20-word vocabulary. The dataset has a high average rate of speech of 102.4 WPM, with many correctly-transcribed sentences having a rate of speech higher than 110 or even 120 WPM. Training a deep convolutional neural network (CNN) with connectionist temporal classification (CTC) and a language model on this dataset results in a 10.7% word error rate. This is a remarkably low error rate for continuous silent speech at over 100 WPM. Calculating the information transfer rate given these metrics results in a bit-rate of 345.8 bits per minute, which is significantly higher than that of other similar systems, and drastically higher than that of BCI systems, which generally report bit-rates between 5 and 25 bits per minute, according to Kronegg et al. [21]. On a smaller, 10-word vocab, 20-sentence dataset, the same model achieves a word error rate of 7.0%, and performs similarly or slightly better on digit sequence prediction benchmarks as well.

This serves to show that continuous silent speech, without any sound or visible movement, can be transcribed and used as a high-accuracy alternative method of communication at rates of speech above 100 WPM. This enables discreet communication when speaking normally is not desirable, as well as communication in environments not suited for normal speech. More importantly, for individuals who have difficulty speaking, such as those with ALS, MS, or other disorders that affect speech, this alternative method of communication could significantly improve the ease and speed in which they communicate.

5.3 Future Work

This system still has much room for improvement. In its current state, the device is somewhat session-dependent, meaning data collected in previous sessions with the device does not necessarily generalize to future sessions. We have found that ensuring the electrodes are placed in exactly the same positions each time can minimize this issue. A larger amount of training data may help to solve this problem as well. The system also is still highly user-dependent, meaning each user must train the system on their own examples of silent speech. In a similar way to how regular speech recognition systems have learned to transcribe speech despite different voices and accents, it is possible that, with enough data, a silent speech recognition system such as this one could learn enough of the variation in silent speech to overcome this issue of user dependence. Alternatively, transfer learning may prove useful in training the system to work with new users without requiring a large volume of training data. Furthermore, we have seen that an increase in vocabulary size or rate of speech, when accompanied by an increase in training data, results in only a slight decrease in accuracy. This is evidence that the system may be able to scale to larger vocabularies or higher rates of speech. An incrementally-trained system, where the user adds training samples each time they want to add a new word to the vocabulary, seems to be a promising way to expand the vocabulary while retaining the system's high accuracy.

There are a number of ways this system could be improved, as well as many other potential applications of this technology. It is my hope that this system will be expanded upon and used as an improved means of communication and information access, helping its users to achieve their goals as a seamless extension of themselves.

Appendix A

Tables

20-Word Vocab, 200-Sentence Dataset			
what am i doing	where the food	are where you	how are you want i
what are you doing the	doing you how	where water are doing what i	am what i the water where
feeling the need food	am cold water	hot water how food cold	how am want hot food cold
hello thirsty hungry want the tired	need i are what you hungry	hello thirsty tired feeling the need	how what where you are feeling
i am doing	cold water where	feeling how cold water	am feeling tired
thirsty hungry are where hot hello	how you doing the hungry	water am tired doing what i	hot need are thirsty hello want
am i the water you need	feeling tired how hot want food	doing what need food	cold feeling tired
food cold feeling hot	want you thirsty hello hungry	am i are the how	tired water want what where food
cold food feeling thirsty	hello am you hungry	doing what i want	are where the feeling how
water where hot food	need doing the hungry hello thirsty	am where i	are what you
tired hello hot water how thirsty	hungry i need you	food are water where the doing	need what am tired cold feeling
feeling cold food	need doing tired how hot	want need thirsty	where hungry hello doing what the
how you are feeling	i am hungry	hungry hot food	want need water
hello are thirsty	doing the cold	how i am where water	want what you need
need food cold feeling	want thirsty tired hungry	hot the feeling hello	what i am doing
where are you cold water	cold food how need	hello tired thirsty hot you need	the water where i are doing
feeling what am tired	thirsty tired cold food	hungry hot how tired	hello where i want
how you are feeling tired the	water tired doing what need	want food cold am hungry	hot hello are thirsty
the food you i want what	feeling cold water	where am doing	thirsty how hot hungry
hello are doing the	want what i	water where you want	am feeling hungry want how
are thirsty hungry hot hello	hot i am doing what	where you want the	hello thirsty you are what need
how am i the hot	feeling cold water tired food	how cold water	cold feeling hungry
am hello where are you thirsty	i want the water where	need doing what	cold feeling how hot
cold food thirsty tired	hello what the hungry	doing are hungry	i am you
where hot water	thirsty need doing feeling how	hello tired you where water	what are the i feeling tired
how am doing	thirsty hot cold food	cold food hot hungry	hello hungry i want you need
are where the water	how am doing what	cold food tired feeling need hot	hot food thirsty want
where i tired the how hello	need are what you hungry	hot food water am doing	hello want are thirsty
i you water where the cold	need what am tired	want cold feeling hungry	how hot food
are thirsty hello	where you doing	what i want the	hello thirsty cold water tired
need are hungry	am where i want	how the food	feeling cold food you how water
hungry thirsty doing what need	where you are hello tired	am what i want	doing the need
need tired feeling cold water am	how tired feeling hungry doing	thirsty feeling you hot water hello	i want the
doing what are how tired	want cold feeling tired hello	hot want the are hungry thirsty	need you i am tired
how am feeling what water	where hot food	doing are hungry thirsty what hello	where the water i am
hello tired doing how you want	feeling hungry where food thirsty	tired i need the hot water	where what are you hungry
tired how cold am doing	hungry cold food hot	are doing hello need the thirsty	how am i need water
you want what thirsty feeling where	hot cold feeling tired	hello doing are food	i am the
need you what	tired cold food feeling how hungry	need food where hot water	i want are thirsty hello
doing what you	feeling the cold water	want hungry how am tired where	hello are the thirsty hot water
you want what	where am tired i want	cold feeling how	hungry thirsty cold food need
hello i the are you hot	what feeling how want water where	am cold food	thirsty need food hot want
hello are hungry	am doing the	how tired i you	need doing cold feeling hello
where what need food	need water want food hot	what you want thirsty need hungry	how the are where am i
thirsty hot food	hello hungry you where i	what are the am	tired food hot water feeling hungry
how hot water	hello need water i thirsty	how what you are tired the	where am feeling tired
tired hot food hungry doing	cold feeling hello	what where am i you thirsty	food are hungry
water the feeling how cold	tired need cold food	doing hello you thirsty want water	am hungry i are the cold
doing what need	how cold food	hello where hot need doing	you want are thirsty
am i doing the how hungry	where what hello need food thirsty	want are hot	what where i need you cold

Table A.1: The 200 sentences generated from the 20 word vocabulary. While some sentences shown are grammatical (eg. "what am i doing", "i am hungry"), most of the sentences are nonsensical, to aid in training the model to learn the alignment between words. Still, there are many phrases contained in the nonsensical sentences that are quite common (eg. "how are you want i") that may actually be used in practice.

Appendix B

Figures

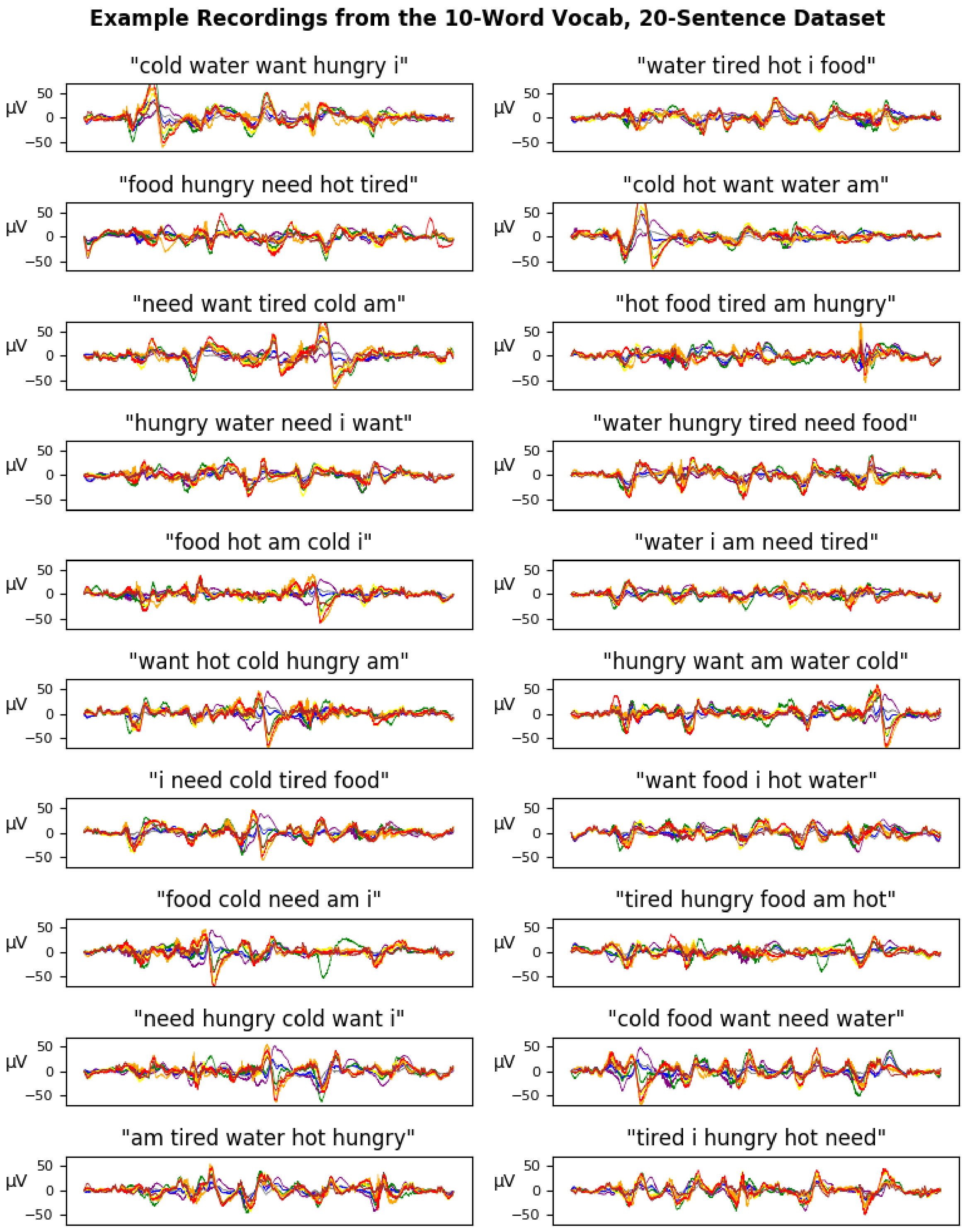


Figure B-1: Examples of signals recorded for each of the 20 sentences in the 10-word vocab dataset.

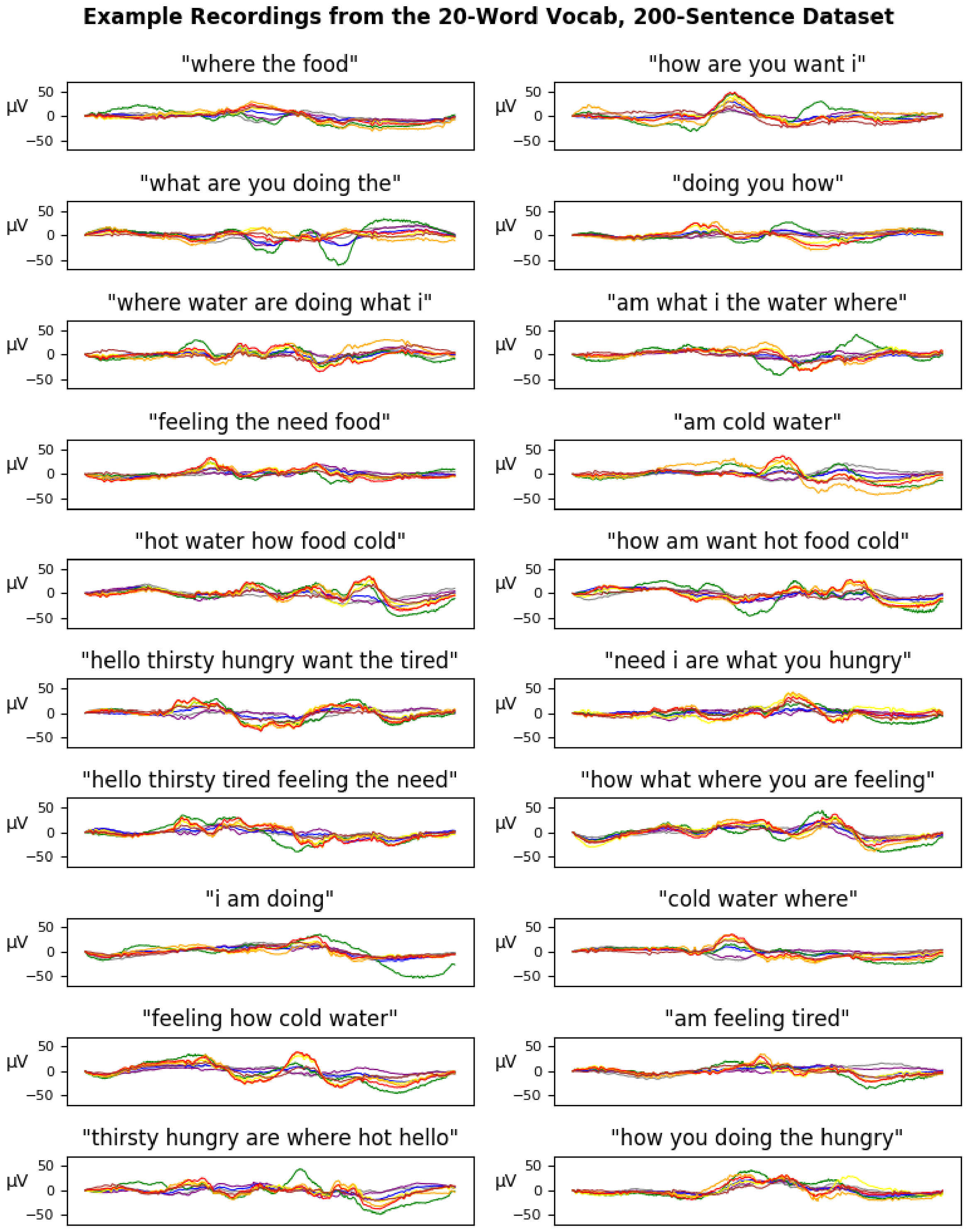


Figure B-2: Examples recordings for 20 of the 200 sentences in the 20-word vocab dataset. The signals shown here have visibly fewer features than the signals of the 10-word vocab dataset (shown in Figure B-1). This is due to the much higher rate of speech of this dataset (over 100 WPM).

Bibliography

[1] Kapur, A., Kapur, S., Maes, P. *AlterEgo: A Personalized Wearable Silent Speech Interface*. In Proceedings of the IUI '18 23rd International Conference on Intelligent User Interfaces. ACM, 43-53. DOI: 10.1145/3172944.3172977. `https://doi.org/10.1145/3172944.3172977`

[2] Kapur, A. (2018) *Human-machine cognitive coalescence through an internal duplex interface*. Massachusetts Institute of Technology. `http://hdl.handle.net/1721.1/120883`

[3] National Institutes of Health, National Institute on Deafness and Other Communication Disorders. *Statistics on Voice, Speech, and Language.* (2019). `https://www.nidcd.nih.gov/health/statistics/statistics-voice-speech-and-language`

[4] Saksamudre, S., Shrishrimal, P.P., Deshmukh, R. *A Review on Different Approaches for Speech Recognition System.* International Journal of Computer Applications. 2015. 115. 23-28. DOI: 10.5120/20284-2839. `https://pdfs.semanticscholar.org/b909/3377c6579b97ab8bd5d4dd9947d372dddc2e.pdf`

[5] Eriksen, C. W., Pollack, M. D., Montague, W. E. (1970). *Implicit speech: Mechanism in perceptual encoding?* Journal of Experimental Psychology, 84(3), 502-507. `http://dx.doi.org/10.1037/h0029274`

[6] Klapp, S. T., Anderson, W. G., Berrian, R. W. (1973). *Implicit speech in reading: Reconsidered*. Journal of Experimental Psychology, 100(2), 368-374. `http://dx.doi.org/10.1037/h0035471`

[7] Jorgensen, C., Lee, D. D., Agabont, S. *Sub auditory speech recognition based on EMG signals*. In Proceedings of the International Joint Conference on Neural Networks, 2003. 2003 Jul 20 (Vol. 4, pp. 3128-3133). IEEE. `https://doi.org/10.1109/IJCNN.2003.1224072`

[8] Brumberg, J. S., Nieto-Castanon, A., Kennedy, P. R., Guenther, F. H. *BrainâĂŞ-computer interfaces for speech communication*. Speech Communication. Volume 52, Issue 4, April 2010, Pages 367-379. `https://doi.org/10.1016/j.specom.2010.01.001`

[9] Anumanchipalli, G. K., Chartier, J., Chang. E. F. (2019). *Speech synthesis from neural decoding of spoken sentences.* Nature 2019. Vol. 568, pages 493âĂŞ498. https://www.nature.com/articles/s41586-019-1119-1

[10] Wand, M., Schultz, T. (2011). *Session-independent EMG-based Speech Recognition.* BIOSIGNALS 2011 - Proceedings of the International Conference on Bio-Inspired Systems and Signal Processing. 295-300. https://www.researchgate.net/publication/221334959

[11] Jou, S., Schultz, T., Walliczek, M., Kraft, F., Waibel, A. (2006). *Towards continuous speech recognition using surface electromyography.* In INTERSPEECH-2006, paper 1592-Mon3WeS.3. https://www.isca-speech.org/archive/interspeech_2006/i06_1592.html

[12] Juang, B. H., Rabiner, L. R. (1991). *Hidden Markov Models for Speech Recognition.* Technometrics, 33:3, 251-272. https://doi.org/10.1080/00401706.1991.10484833

[13] Lu, L., Zhang, X., Renais, S. *On training the recurrent neural network encoder-decoder for large vocabulary end-to-end speech recognition.* 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Shanghai, 2016, pp. 5060-5064. https://doi.org/10.1109/ICASSP.2016.7472641

[14] Pascanu, R., Mikolov, T., Bengio, Y. (2013). *On the difficulty of training recurrent neural networks.* ICML'13 Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28. Pages III-1310-III-1318. https://dl.acm.org/citation.cfm?id=3043083

[15] Mewett, D. T., Nazeran, H., Reynolds, K. J. *Removing power line noise from recorded EMG.* 2001 Conference Proceedings of the 23rd Annual International Conference of the IEEE Engineering in Medicine and Biology Society, Istanbul, Turkey, 2001, pp. 2190-2193 vol.3. https://doi.org/10.1109/IEMBS.2001.1017205

[16] Graves, A., Fernandez, S., Gomez, F., Schmidhuber, J. *Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks.* ICML '06 Proceedings of the 23rd international conference on Machine learning. 2006. Pages 369-376. https://doi.org/10.1145/1143844.1143891

[17] Bourlard H., Morgan N. *Hybrid HMM/ANN systems for speech recognition: Overview and new research directions.* In: Giles C.L., Gori M. (eds) Adaptive Processing of Sequences and Data Structures. NN 1997. Lecture Notes in Computer Science, vol 1387. Springer, Berlin, Heidelberg. https://doi.org/10.1007/bfb0054006

[18] Jelinek, F. *Continuous speech recognition by statistical methods.* Proceedings if the IEEE, Vol. 64, NO. 4, April 1976. https://doi.org/10.1109/PROC.1976.10159

[19] Chow, Y., Schwartz, R. *The N-Best algorithm: an efficient procedure for finding top N sentence hypotheses*. HLT '89 Proceedings of the workshop on Speech and Natural Language Pages 199-202. `https://doi.org/10.3115/1075434.1075467`

[20] Wand, M., Janke, M., Schultz, T. *The EMG-UKA corpus for electromyographic speech processing*. In The 15th Annual Conference of the International Speech Communication Association. Interspeech 2014. `https://www.semanticscholar.org/paper/3cc02344bb7e13ef04dd4540ea967b463c02e488`

[21] Kronegg, J., Voloshynovskyy, S., Pun, T. *Information-transfer rate modeling of EEG-based synchronized brain-computer interfaces*. GenÃÍve, 2005. `https://archive-ouverte.unige.ch/unige:48013`

[22] Wolpaw, J. R., Ramoser, H., McFarland. D. J., Pfurtscheller, G. *EEG-Based Communication: Improved Accuracy by Response Verification*. IEEE Trans. Rehab. Eng., Vol. 6, pp. 326-333, 1998.